

केन्द्रीय विद्यालय संगठन
Kendriya Vidyalaya Sangathan



STUDY MATERIAL
(Computer Science)

CLASS-XII
2014-15

KENDRIYA VIDYALAYA SANGATHAN
GURGAON REGION
SECTOR-14, OLD DELHI GURGAON ROAD, GURGAON
(HARYANA)- 122001

STUDY MATERIAL
CLASS XII (COMPUTER SCIENCE)

CHIEF PATRON:
Sh. AVINASH DIKSHIT
(COMMISSIONER, KVS)

PATRON:
MR. C. MANI
(DEPUTY COMMISSIONER, GURGAON REGION)

GUIDE:
Dr. A. K. SHARMA, ASSISTANT COMMISSIONER, GURGAON REGION
Sh. B.L.MORODIA, ASSISTANT COMMISSIONER, GURGAON REGION
Sh. C.S AZAD, ASSISTANT COMMISSIONER, GURGAON REGION

CORDINATOR:
MRS. RAJNI H. UPPAL
PRINCIPAL KV, SEC. 8. ROHINI, NEW DELHI

SUBJECT CONTRIBUTORS:-
Mr. Lavendra Kumar Tyagi, PGT (Comp. Sc.) K. V. Sec. 8 Rohini, New Delhi
Mr. Omprakash, PGT (Comp. Sc.) K. V. Sec. 8 Rohini, New Delhi
Mr. Bhupesh Bhatt, PGT (Comp. Sc.) K. V. AFS Rajokri, New Delhi
Mr. Amit Saxena, PGT (Comp. Sc.) K. V. Sec. 3 Rohini , New Delhi
Mrs. Neelima , PGT (Comp. Sc.) K. V. Sec. 3, Rohini, New Delhi
Mrs. Bhawana Duggal, PGT (Comp. Sc.) K. V. Sec. 22, Rohini, New Delhi
Mr. Manoj Kulshrestha, PGT (Comp. Sc.) K. V. AFS Bawana, New Delhi

INDEX

Sr. No.	Contents	Page No.
1	Syllabus	4-6
2	Unit-1: Object Oriented Programming in C++	7-90
3	Unit-2: Data Structure	91-110
4	Unit-3: Database Management System & SQL	111-123
5	Unit-4: Boolean Algebra	124-136
6	Unit-5 Networking & Open Source Software	137-161
7	Sample Question Paper	162-177

Computer Science (083)

Class XII (Theory) - C++

Duration: 3 hours

Total Marks: 70

Unit No.	Unit Name	Marks
1.	OBJECT ORIENTED PROGRAMMING IN C++	30
2.	DATA STRUCTURE	14
3.	DATABASE MANAGEMENT SYSTEM AND SQL	8
4.	BOOLEAN ALGEBRA	8
5.	COMMUNICATION TECHNOLOGIES	10
		70

UNIT 1: OBJECT ORIENTED PROGRAMMING IN C++ (50 Theory + 40 Practical) Periods

REVIEW: C++ covered in Class - XI,

Object Oriented Programming: Concept of Object Oriented Programming - Data hiding, Data encapsulation,

Class and Object, Abstract class and Concrete class, Polymorphism (Implementation of polymorphism using Function overloading as an example in C++); Inheritance, Advantages of Object Oriented Programming over earlier programming methodologies,

Implementation of Object Oriented Programming concepts in C++: Definition of a class, Member of a class - Data Members and Member Functions (methods), Using Private and Public visibility modes, default visibility mode (private); Member function definition: inside class definition and outside class definition using scope resolution operator (::); Declaration of objects as instances of a class; accessing members from object (s), Objects as function arguments - pass by value and pass by reference;

Constructor and Destructor: Constructor: special characteristics, declaration and definition of a constructor, default constructor, overloaded constructors, copy constructor, constructor with default arguments;

Destructor: Special Characteristics, declaration and definition of destructor;

Inheritance (Extending Classes) : Concept of Inheritance, Base Class, Derived classes, protected visibility mode; Single level inheritance, Multilevel inheritance and Multiple inheritance, Privately derived, publicly derived and Protectedly derived class, accessibility of members from objects and within derived class (es);

Data File Handling: Need for a data file, Types of data files - Text file and Binary file;

Text File: Basic file operations on text file: Creating/Writing text into file, Reading and Manipulation of text from an already existing text file (accessing sequentially);

Binary File: Creation of file, Writing data into file, Searching for required data from file, Appending data to a file, Insertion of data in sorted file, Deletion of data from file, Modification of data in a file; Implementation of above mentioned data file handling in C++;

Components of C++ to be used with file handling:

Header file: fstream.h; ifstream, ofstream, fstream classes;

Opening a file in in, out, and app modes;

Using cascading operators (>><<) for writing text to the file and reading text from the file; open(), get(), read(), put(), write(), getline() and close() functions; Detecting end-of-file (with or without using eof() function), tellg(), tellp(), seekg(), seekp()

Pointers:

Introduction to Pointer, Declaration and Initialization of Pointer; Dynamic memory allocation/deallocation operators: new, delete; Pointers and Arrays: Array of Pointers, Pointer to an array (1 dimensional array), Function returning a pointer, Pointer to structure: De-reference operator; self referential structure;

UNIT 2: DATA STRUCTURES (42 Theory + 36 Practical) Periods

Introduction to data structure – arrays, stacks, queues

Arrays: One and two Dimensional arrays: Sequential allocation and address calculation;

One dimensional array: Traversal, Searching (Linear, Binary Search), Insertion of an element in an array, deletion of an element from an array, Sorting (Insertion, Selection, Bubble)

Two-dimensional arrays: Traversal Finding sum/difference of two NxM arrays containing numeric values, Interchanging Row and Column elements in a two dimensional array;

Stack (Array and Linked implementation of Stack): Introduction to stack (LIFO_Last in First Out Operations) Operations on Stack (PUSH and POP) and its Implementation in C++, Converting expressions from INFIX to POSTFIX notation and evaluation of Postfix expression;

Queue: (Circular Array and Linked Implementation): Introduction to Queue (FIFO - First in First out operations) Operations on Queue (Insert and Delete and its Implementation in C++.

UNIT 3: DATABASES AND SQL (20 Theory + 20 Practical) Periods

Data base Concepts: Introduction to data base concepts and its need.

Relational data model: Concept of domain, tuple, relation, key, primary key, alternate key, candidate key;

Relational algebra: Selection, Projection, Union and Cartesian product;

Structured Query Language:

General Concepts: Advantages of using SQL, Data Definition Language and Data Manipulation Language;

Data Types: NUMBER/DECIMAL, CHARACTER/VARCHAR/VARCHAR2, DATE; SQL COMMANDS: CREATE TABLE, DROP TABLE, ALTER TABLE, UPDATESET....., INSERT, DELETE; SELECT, DISTINCT, FROM, WHERE, IN, BETWEEN, GROUPBY, HAVING, ORDERBY;

SQL functions: SUM, AVG, COUNT, MAX AND MIN;

Obtaining results (SELECT query) from 2 tables using equi-join and Union

Note: Implementation of the above mentioned commands could be done on any SQL supported software

UNIT 4: BOOLEAN ALGEBRA (16 Theory + 0 Practical) Periods

Role of Logical Operations in Computing.

Binary-valued Quantities, Boolean Variable, Boolean Constant and Boolean Operators: AND, OR, NOT; **Truth Tables**; Closure Property, Commutative Law, Associative Law, Identity law, Inverse Law, Principle of Duality, Idem potent Law, Distributive Law, Absorption Law, Involution Law, DeMorgan's Law and their applications;

Obtaining Sum of Product (SOP) and Product of Sum (POS) form from the Truth Table, Reducing Boolean

Expression (SOP and POS) to its minimal form, Use of Karnaugh Map for minimization of Boolean expressions (up to 4 variables);

Application of Boolean Logic: Digital electronic circuit design using basic Logic Gates (NOT, AND, OR, NAND, NOR)

Use of Boolean operators (AND, OR) in search engine queries.

UNIT 5: COMMUNICATION TECHNOLOGIES (16 Theory + 0 Practical) Periods

Evolution of Networking: ARPANET, Internet, Interspace

Different ways of sending data across the network with reference to switching techniques (Circuit and Packet switching);

Data Communication terminologies: Concept of Channel, Bandwidth (Hz, KHz, MHz) and Data transfer rate (bps, kbps, Mbps, Gbps, Tbps);

Transmission media: Twisted pair cable, coaxial cable, optical fiber, infrared, radio link, microwave link and satellite link;

Network devices: Modem, RJ45 connector, Ethernet Card, Router, Repeater, Switch, Gateway, wifi card;

Network Topologies and types: Bus, Star, Tree, PAN, LAN, WAN, MAN;

Network Protocol: TCP/IP, File Transfer Protocol (FTP), PPP, SMTP, POP3, Remote Login (Telnet), Internet Wireless/Mobile Communication protocol such as GSM, CDMA, GPRS, WLL,

Mobile Telecommunication Technologies: 1G, 2G, 3G and 4G

Protocols for Chat and Video Conferencing VOIP

Wireless technologies such as Wi-Fi and WiMax

Network Security Concepts: Threats and prevention from Viruses, Worms, Trojan horse, Spams

Use of Cookies, Protection using Firewall;

India IT Act, Cyber Law, Cyber Crimes, IPR issues, Hacking;

Introduction To Web services: WWW, Hyper Text Markup Language (HTML), eXtensible Markup Language (XML); Hyper Text Transfer Protocol (HTTP); Domain Names; URL; Website, Web browser, Web Servers; Web Hosting, Web Scripting - Client side (VB Script, Java Script, PHP) and Server side (ASP, JSP, PHP), Web 2.0 (for social networking)

Unit-I

Objective Oriented Programming in C++

Review of C++:-

C++ Character Sets:

Letters	A-Z , a-z
Digits	0-9
Special Symbols	Space + - * / ^ \ () [] { } = != < > . , " \$, ; : % ! & ? _ # <= >= @
White Spaces	Blank spaces, horizontal tab, carriage return
Other Characters	Any of the 256 ASCII character

Token:-The smallest individual unit in a program is known as token. Tokens used in C++ are:

KEYWORDS :

Keywords are the certain reserved words that convey a special meaning to the compiler. These are reserve for special purpose and must not be used as identifier name.eg for , if, else , this , do, etc.

IDENTIFIERS:

Identifiers are programmer defined names given to the various program elements such as variables, functions, arrays, objects, classes, etc.. It may contain digits, letters and underscore, and must begin with a letter or underscore. C++ is case sensitive as it treats upper and lower case letters differently. The following are some valid identifiers:

Pen time580 s2e2r3 _dos _HJI3_JK

LITERALS:

The data items which never change their value throughout the program run. There are several kinds of literals:

- Integer literals
- Character literals
- Floating literals
- String literals

Integer literals :

Integer literals are whole numbers without any fractional part. An integer literal must have at least one digit and must not contain any decimal point. It may contain either + or - sign. A number with no sign is assumed as positive. C++ allows three types of integer literals:

- (i) Decimal Integer Literals:- An integer literal without leading 0 (zero) is called decimal integer literals e.g., 123, 786 , +97 , etc.
- (ii) Octal Integer Literals:- A sequence of octal digit starting with 0 (zero) is taken to be an octal integer literal (zero followed by octal digits). e.g., 0345, 0123 , etc.
- (iii) Hexadecimal Integer Literals :- Hexadecimal Integer Literals starts with 0x or 0X followed by any hexa digits. e.g., 0x9A45, 0X1234, etc.

Character literals:

Any single character enclosed within single quotes is a character literal.

e.g 'A' , '3'

Floating literals:

Numbers which are having the fractional part are referred as floating literals or real literals. It may be a positive or negative number. A number with no sign is assumed to be a positive number.

e.g 2.0, 17.5, -0.00256

String Literals:

It is a sequence of character surrounded by double quotes. e.g., "abc", "23".

PUNCTUATORS:

The following characters are used as punctuators which are also known as separators in C++

[] { } () , ; : * = #

Punctuator	Name	Function
[]	Brackets	These indicates single and multidimensional array subscripts
()	Parenthesis	These indicate function calls and function parameters.
{ }	Braces	Indicate the start and end of compound statements.
;	Semicolon	This is a statement terminator.
,	Comma	It is used as a separator.
:	Colon	It indicates a labeled statement
*	Asterisk	It is used as a pointer declaration
...	Ellipsis	These are used in the formal argument lists of function prototype to indicate a variable number of arguments.
=	Equal to	It is used as an assigning operator.
#	Pound sign	This is used as preprocessor directives.

OPERATORS:

An operator is a symbol or character or word which trigger some operation (computation) on its operands.

(i) **Unary operators:** Those which require only one operand to operate upon. e.g. unary -, unary +, ++, -- !.

(ii) **Binary operators:** Binary operators require two operands to operate upon. e.g. +, *, /, -, etc.

(iii) **Ternary Operator :** Ternary operator require three operands to operate upon.

Conditional operator (? :) is a ternary operator in C++.

Structure of a C++ program:

Structure of a C++ program	A C++ program to prints a string on the screen
<pre>#include<iostream.h> void main () { Statement_1; Statement_2; : : }</pre>	<pre>#include<iostream.h> void main () { cout<< "Kendriya Vidyalaya"; } The program produces following output: Kendriya Vidyalaya</pre>

COMMENTS IN A C++ PROGRAM.:

Comments are the pieces of code that compiler ignores to compile. There are two types of comments in C++.

1. **Single line comment:** The comments that begin with // are single line comments. The Compiler simply ignores everything following // in the same line.
2. **Multiline Comment :** The multiline comment begin with /* and end with */ . This means everything that falls between /* and */ is consider a comment even though it is spread across many lines.

Input Output (I/O) operations In C++: Input & Output operations are supported by the istream (input stream) and ostream (output stream) classes. The predefined stream objects for input, output are :

(i) The cout Object:

The identifier cout is a predefined object of ostream class that represents the standered output stream in C++ and tied to slandered output. cout stands for console output . cout sends all output to the standard output device i.e. monitor.

The syntax of **cout** is as follows:

```
cout<< data;
```

Where data may be a variable or constant or string etc. e.g.

```
cout<< a ;           ( here a can be any variable)
```

Output Operator (<<): The output operator (<<) is also known as 'stream insertion' or 'put to' operator. It directs the contents of the variable (or value) on its right to the object on its left (i.e., cout).

(ii) The cin Object :

The cin object is an istream class object tied to slandered input. cin stands for console input.

cin object used to get input from the keyboard. When a program reaches the line with cin, the user at the keyboard can enter values directly into variables.

The syntax of **cin** is as follows:

```
cin>> variablename;
```

e.g

```
cin>> ch;           ( here ch can be any variable)
```

input Operator (>>): The input operator (>>) is also known as extraction or ‘get from’ operator . It extracts (or takes) the value from the keyboard and assign it to the variable on its right.

CASCADING OF OPERATOR:

When input or output (>> or <<) are used more than one time in a single statement then it is called as cascading of operators.

e.g `cout<< roll<< age<< endl;`

DATA TYPES IN C++:

Data types are means to identify the types of data and associated operations of handling it. Data types in C++ are of two types:

1. Fundamental or Built-in data types .
2. Derived data types.

1. Fundamental or Built-in data types: These data types are already known to compiler. These are the data types those are not composed of other data types. There are following fundamental data types in C++:

- (i) **int data type (for integer) :-** int data type is used for integer value. An identifiers declare as int cannot have fractional part.
- (iii) **char data type (for characters):-** An identifiers declare as char can store a character.
- (iv) **float data type (for floating point numbers):-** An identifier declare as float can hold a floating point number.
- (v) **double data type (for double precision floating point numbers):-** The double data type is also used for handling floating point numbers but it occupies twice as much memory as float and store numbers with much larger range and precision.

Data Type Modifiers:- There are following four data type modifiers in C++ , which may be used to modify the fundamental datatypes to fit various situations more precisely:

- (i) signed
- (ii) unsigned
- (iii) long
- (iv) short

signed, unsigned, long, short data type modifiers may be apply to char & int data types. However you may also apply long to double

Data Type	Size (in Bytes)	Range
char	1	-128 to 127
unsigned char	1	0 to 255
Signed char	1	Same as char
Int	2	-32768 to 32767
unsigned int	2	0 to 65535
signed int	2	Same as int
short (or short int)	2	-32768 to 32767
long (or long int)	4	-2147483648 to 2147483647
float	4	3.4×10^{-38} to $3.4 \times 10^{38} - 1$ (upto 7 digits of precision)
double	8	1.7×10^{-308} to $1.7 \times 10^{308} - 1$ (upto 15 digits of precision)
long double	10	3.4×10^{-4932} to $1.1 \times 10^{4932} - 1$ (upto 19 digits of precision)

2. Derived Data Types:- These are the data types that are composed of fundamental data types. e.g., array, class, structure, etc.

Variables:-A named memory location, whose contains can be changed with in program execution is known as variable. OR

A variable is an identifier that denotes a storage location, which contains can be varied during program execution.

Declaration of Variables:- All variables must be declared before they are used in executable statements. Variable declaration reserves memory required for data storage and associates it with a name. Syntax for variable declaration is:

```
datatypes variable_name1, variable_name2, variable_name3,..... ;
```

e.g.,

```
int num;  
int num, sum, avg;
```

We can also initialize a variable at the time of declaration by using following syntax:

```
datatypes variable_name = value;
```

e.g.,

```
int num = 0;
```

Constant:- A named memory location, whose contains cannot be changed with in program execution is known as constant. OR

A constant is an identifier that denotes a storage location, which contains cannot be varied during program execution.

Syntax for constant declaration is:

```
const datatypes constant_name = value ;
```

e.g.,

```
const float pi = 3,14f ;
```

Formatted Output (Manipulators) :

Manipulators are the operators used with the insertion operator << to format the data display. The most commonly used manipulators are **endl** and **setw**.

1. **The endl manipulator :**The endl manipulator, when used in a output statement , causes a line feed to be inserted. It has same effect as using new line character “\n”. e.g.,

```
cout<< “ Kendriya Vidyalaya Sangathan”<<endl;  
cout<< “ Human Resource and Development”;
```

The output of the above code will be

```
Kendriya Vidyalaya Sangathan  
Human Resource and development
```

2. **The setw() Manipulator :** The **setw()** manipulator sets the width of the field assign for the output. It takes the size of the field (in number of character) as a parameter. The output will be right justified e.g., the code :

```
cout<<setw(6)<<”R” ;
```

Generates the following output on the screen (each underscore represent a blank space)

```
_____R
```

In order to use these manipulator , it is must to include header file `iomanip.h`

Operators:-

Arithmetic operators :- Those operators are operates only on numeric data types operands are known as arithmetic operators.

Operator	Operation	Example
Unary -	Result is the negation of operand's value (Reverse the sign of operand's value)	If a=5 then - a means -5. If a = - 4 then - a means 4.
Unary +	The result is the value of its operand	If a=5 then +a means 5. If a = - 4 then +a means -4.
+ (Addition Operator)	Addition (it adds two numbers)	4+20 results is 24. If a = 5 then a + 5 results 10.
- (Subtraction Operator)	Subtraction (Subtract the second operand from first)	14 - 3 evaluates to 12. If a = 2 , b = 3 then b - a evaluates 1.
* (Multiplication Operator)	Multiplies the values of its operands	3*4 evaluates to 12. If a=2, b=3 then a*b evaluates to 6.
/ (Division Operator)	Divides its first operand by the second	100/5 evaluates 20. If a =10 , b = 5 then a/b evaluates 2
% (Modulus Operator)	It produce the remainder of dividing the first operand by second	19%6 evaluates to 1. If a = 14 , b = 3 then a%b evaluates to 2. Modulus operator requires that both operands be integer and second operand be non-zero.

Increment and Decrement Operators (++ , - -) :

The increment operator (++) adds 1 to its operand and decrement operator (--) subtract one from its operand. In other word

a = a + 1; is same as ++a; or a++;

& a = a - 1 ; is same as --a; or a--;

Both the increment & decrement operators comes in two version :

- (i) Prefix increment/decrement :- When an increment or decrement operator precedes its operand, it is called prefix increment or decrement (or pre-increment / decrement). In prefix increment/decrement , C++ perform the increment or decrement operation before using the value of the operand. e.g.,
If sum = 10 and count =10 then
Sum = sum (++count);
First count incremented and then evaluate sum = 21.
- (ii) Postfix increment/decrement :- When an increment or decrement operator follows its operand, it is called postfix increment or decrement (or post-increment / decrement). In postfix increment/decrement , C++ first uses the value of the operand in evaluating the expression before incrementing or decrementing the operand's value. e.g.,
If sum = 10 and count =10 then
Sum = sum +(count++);
First evaluate sum = 20 , and then increment count to 11.

Relational Operator: These operators are used to compare two values. If comparison is true, the relational expression results into the value 1 and if the comparison is false its result will be 0. The six relational operators are:

Operator	Meaning
==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Logical Operators : In addition to the relational operator, C++ contains three logical operators. Relational operators often are used with logical operators to construct more complex decision making expressions.

Operators	Use	Return True if
&& (Logical AND)	op1 && op2	op1 and op2 are both true
(Logical OR)	op1 op2	Either op1 or op2 is true
! (Logical NOT)	!op1	op1 is false (it is unary operator)

Assignment Operator: C++ offers an assignment operator (=) to assign a value to an identifier. The assignment statement that make use of this operator are written in the form :

var = expression ;

where var generally represents a variable and expression may be a constant or a variable or an expression.

C++ offers special shorthand operators that simplify the coding of a certain type of assignment statement . e.g.,

a = a + 10 ; can be written as a+=10 ;

This shorthand works for all binary arithmetic operators. The general form of this shorthand is

Var = var operator expression ; is same as
var operator = expression ;

Following are some examples of C++ shorthands:

x -=10 ;	equivalent to	x = x -10 ;
x *=3 ;	equivalent to	x = x * 3 ;
x /=2 ;	equivalent to	x = x/2 ;
x %=z	equivalent to	x = x % z ;

Conditional operator (? :)

The conditional operator (? :) is a ternary operator i.e., it require three operands. The general form of conditional operator is:

expression1? expression2: expression3 ;

Where expression1 is a logical expression , which is either true or false.

If expression1 evaluates to true i.e., 1, then the value of whole expression is the value of expression2, otherwise, the value of the whole expression is the value of expression3. For example

```
min = a < b ? a : b ;
```

Here if expression (a < b) is true then the value of a will be assigned to min otherwise value of b will be assigned to min.

Comma operator (,)

The comma operator (,) is used to separate two or more expressions that are included where only one expression is expected. When the set of expressions has to be evaluated for a value, only the rightmost expression is considered.

For example, the following code:

```
a = ( b = 3 , b + 2 );
```

Would first assign the value 3 to b, and then assign b+2 to variable a. So, at the end, variable a would contain the value 5 while variable b would contain value 3.

sizeof()

This operator returns the size of its operand in bytes. The operand may be an expression or identifier or it may be a data type.

```
a = sizeof (char);
```

This will assign the value 1 to a because char is a one-byte long type.

Expressions:-An expression in C++ is any valid combination of operators, constants, and variables.

Pure Expressions:-If an expression have all operand of same data types then it is called a pure expression.

Mixed Expressions :-If an expression have operands of two or more different data types then it is called a mixed expression.

Arithmetic Expressions:-Arithmetic expression can either be integer expressions or real expressions. Sometimes a mixed expression can also be formed which is a mixture of real and integer expressions.

Integer Expressions:- Integer expressions are formed by connecting all integer operands using integer arithmetic operators.

Real Expressions:- Real expressions are formed by connecting real operands by using real arithmetic operators.

Logical Expressions:- The expressions which results evaluates either 0 (false) or 1 (true) are called logical expressions. The logical expressions use relational or Logical operators.

Type Conversion:-The process of converting one predefined data type into another is called type conversion.

C++ facilitates the type conversion in two forms:

- (i) **Implicit type conversion:-** An implicit type conversion is a conversion performed by the compiler without programmer's intervention. An implicit conversion is applied generally whenever different data types are intermixed in an expression. The C++ compiler converts all operands upto the data type of the largest data type's operand, which is called type promotion.
- (ii) **Explicit type conversion :-** An explicit type conversion is user-defined that forces an expression to be of specific data type.

Type Casting:- The explicit conversion of an operand to a specific type is called type casting.

Type Casting Operator - (type) :-Type casting operators allow you to convert a data item of a given type to another data type. To do so, the expression or identifier must be preceded by the name of the desired data type, enclosed in parentheses. i. e.,

(data type) expression

Where data type is a valid C++ data type to which the conversion is to be done. For example, to make sure that the expression $(x+y/2)$ evaluates to type float, write it as:

(float) $(x+y/2)$

Precedence of Operators:- Operator precedence determines which operator will be performed first in a group of operators with different precedence. For instance $5 + 3 * 2$ is calculated as $5 + (3 * 2)$, giving 11

Statements:-Statements are the instructions given to the Computer to perform any kind of action.

Null Statement:-A null statement is useful in those case where syntax of the language requires the presence of a statement but logic of program does not give permission to do anything then we can use null statement. A null statement is nothing only a ;. A null (or empty statement) have the following form:

;

// only a semicolon (;)

Compound Statement :-A compound statement is a group of statements enclosed in the braces { }. A Compound statement is useful in those case where syntax of the language requires the presence of only one statement but logic of program have to do more thing i.e., we want to give more than one statement in place of one statement then we can use compound statement.

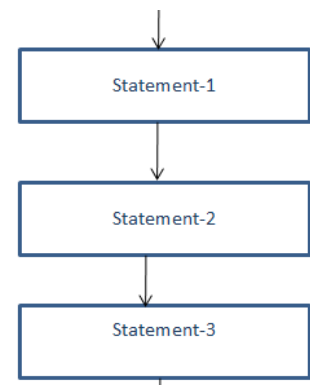
```
{
    St-1;
    St-2;
    :
    :
}
```

Statement Flow Control:-In a program, statements may be executed sequentially, selectively, or iteratively.

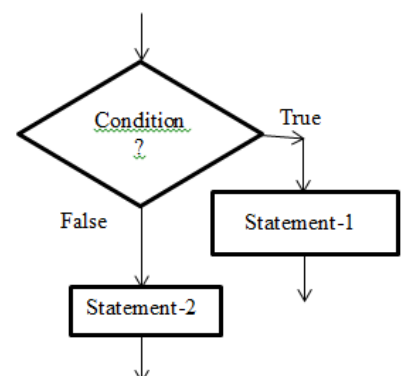
Every programming language provides three constructs:

1. Sequence Constructs
2. Selection Constructs
3. Iteration Constructs

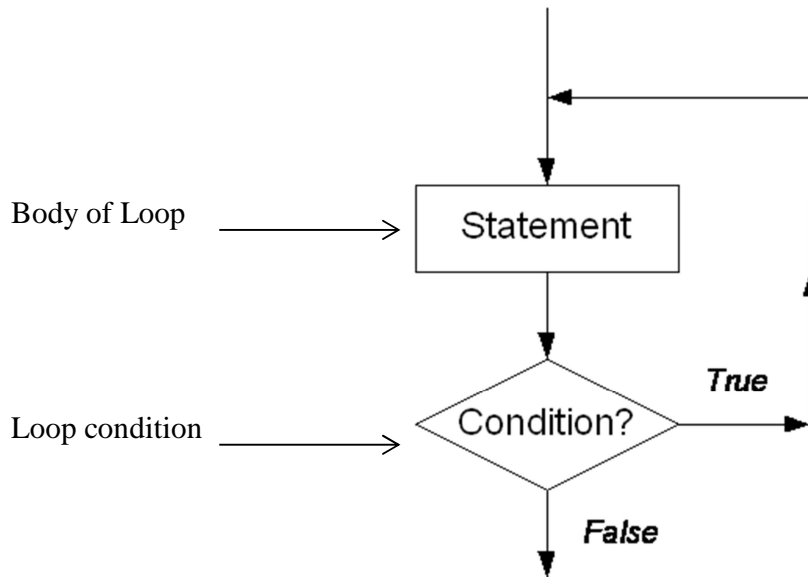
Sequence Construct:-The sequence construct means the statements are being executed sequentially. It represents the default flow of statements.



Selection Construct:- The selection construct means the execution of statement(s) depending on a condition. If a condition is true, a group of statements will be execute otherwise another group of statements will be execute.

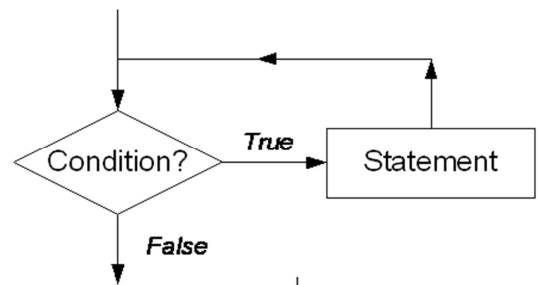


Looping or Iteration Statements:- Looping the iteration construct means repetition of set of statements depending upon a condition test. The iteration statements allow a set of instructions to be performed repeatedly until a certain condition is true.

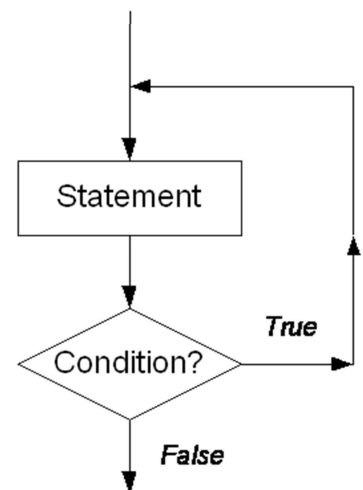


There are two types of loops:-

1. Entry-controlled loop :- In entry-controlled loop first of all loop condition is checked and then body of loop is executed if condition is true. If loop condition is false in the starting the body of loop is not executed even once.



2. Exit-controlled loop :- In exit-controlled loop first body of loop is executed once and then loop condition is checked. If condition is true then the body of loop will be executed again. It means in this type of loop, loop body will be executed once without checking loop condition.



Selection Statements :- There are two types of selection statements in C++ :

1. if statement
2. switch statement

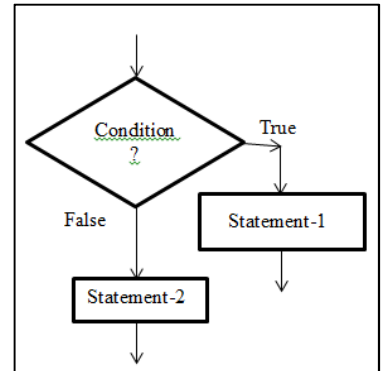
1. if Statement : If statement have three forms

(a) if ... else Statement :- It is useable, when we have to performs an action if a condition is True and we have to perform a different action if the condition is false. The syntax of if...else statement is:

```

if ( < conditional expression > )
{
< statement-1 or block-1>;
    // statements to be executed when conditional expression is true.
}
else
{
< statement-2 or block-2>;
    // statements to be executed when conditional expression is false.
}

```



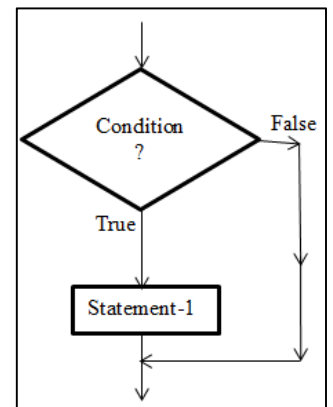
If the <conditional expression> is evaluated to true then the < statement-1 or block-1> (statement under if () block) will be executed otherwise the <statement-2 or block-2> (statements under else block) would be executed. if there exists only one program statement under if() block then we may omit curly braces { }.

(b) Simple if statement:- The else part in if ... else statement is optional, if we omit the else part then it becomes simple if statement. This statement is usable, when we have to either perform an action if a condition is True or skips the action if the condition is false. The syntax of simple if statement is:

```

if ( < conditional expression > )
{
    < statement-1 or block-1>;
    // statements to be executed when conditional expression is true.
}

```



Here <statement-1 or block-1> will be executed only if <conditional expression > evaluates true. if there exists only one program statement under if() block then we may omit curly braces { }.

(c) The if-else-if ladder :-This statement allows you to test a number of mutually exclusive cases and only execute one set of statements for which condition evaluates true first.

The syntax is:

```

if ( <condition -1> )
    statement-1;    // do something if condition-1 is satisfied (True)
else if ( <condition - 2 >)
    statement-3 ;  // do something if condition -2 is satisfied (True)
else if (<condition - 3 >)
    statement-3 ;    // do something if condition- 3 is satisfied (True)
    :
    : // many more n-1 else - if ladder may come
    :
else if( < condition - n >)
    statement-n ;    // do something if condition - n is satisfied (True)
else
    statement-m ;    // at last do here something when none of the
                    // above conditions gets satisfied (True)
}

```

<> in syntax is known as a place holder, it is not a part of syntax, do not type it while writing program. It only signifies that anything being kept there varies from program to program.
 [] is also not a part of syntax, it is used to mark optional part of syntax i.e. all part of syntax between [] is optional.

In the above syntax there are ladder of multiple conditions presented by each if(), all of these conditions are mutually exclusive. If one of them would evaluates true then the statement followed that condition will be executed and all the conditions below it would not be evaluated (checked).

Say suppose if condition-3 gets satisfy (i.e. evaluates true value for the condition), then statement-3 gets executed and all other conditions below it would be discarded.

If none of the n if () conditions gets satisfied then the last else part always gets executed. It is not compulsory to add an else at the last of the ladder.

We can also write a group of statement enclosed in curly braces { } (as a compound statement) in place of any statement (statement-1. Statement-2,....., statement-n) if required in above syntax.

Nested if Statement:- If an if statement is written in the if or else clause of another if statement then it is known as nested if. Some possible syntax of nested if statements given below:

Syntax 1:-

```
if ( <outer- condition > )
{
    if ( <inner-condition> )
    {

        //some statements to be executed
        // on satisfaction of inner if ( ) condition.

    } // end of scope of inner if( )
    //some statements to be executed
    // on satisfaction of outer if ( ) condition.

} // end of the scope of outer if( )
```

Syntax 2:-

```
if ( <outer- condition > )
{
    if ( <inner-condition> )
    {

        //some statements to be executed
        // on satisfaction of inner if ( ) condition.

    }
    else
    {
        // statements on failure of inner if( )
    }
}
```

```

        //some statements to be executed
        // on satisfaction of outer if ( ) condition.
    }
else
{

        // statements on failure of outer if( )
}

```

2. switch Statement :-This is multi-branching statement. Syntax of this statement is as follows:

```

switch (expression/variable)
{
    case value_1: statement -1;
                break;
    case value_2: statement -2;
                break;
    :
    :
    case value_n: statement -n;
                break;
    [ default: statement -m ]
}

```

Note: expression/variable should be integer or character type only.

When the switch statement is executed, the expression/variable is evaluated and control is transferred directly to the statement whose case label value matches the value of expression/variable. If none of the case label value matches the value of expression/variable then only the statement following the default will be executed. If no default statement is there and no match is found then no action take place. In this case control is transferred to the statement that follows the switch statement.

Loops in C++:-There are three loops or iteration statements are available in C++

1. for loop
2. while loop
3. do.... while loop

1. **The for Loop:**For loop is a entry control loop the syntax of for loop is :

```

for(initialization_expression(s); loop_Condition; update_expression)
{
    Body of loop
}

```

Working of the for Loop:-

1. The *initialization_expression* is executed once, before anything else in the for loop.
2. The *loop condition* is executed before the body of the loop.
3. If loop condition is true then body of loop will be executed.
4. The *update expression* is executed after the body of the loop

- After the *update expression* is executed, we go back and test the *loop condition* again, if *loop_condition* is true then body of loop will be executed again, and it will be continue until *loop_condition* becomes false.

Example:

```
for (int i = 0; i < 7; i++)
    cout<<i * i << endl;
```

Interpretation:

An int i is declared for the duration of the loop and its value initialized to 0. i² is output in the body of the loop and then i is incremented. This continues until i is 7.

- while Loop:-** while loop is also an entry controlled loop. The syntax of while loop is :

```
while (loop_condition)
{
    Loop_body
}
```

Where the Loop_body may contain a single statement, a compound statement or an empty statement.

The loop iterates (Repeatedly execute) while the loop_condition evaluates to true. When the loop_condition becomes false, the program control passes to the statement after the loop_body.



In while loop , a loop control variable should be initialized before the loops begins. The loop variable should be updated inside the loop_body.

- do-while loop:-** do-while loop is an exit-controlled loop i.e. it evaluates its loop_condition at the bottom of the loop after executing its loop_body statements. It means that a do-while loop always executes at least once. The syntax of do-while loop is:

```
do
{
    Loop_body
}while (loop_condition);
```

In do-while loop first of all loop_body will be executed and then loop_condition will be evaluates if loop_condition is true then loop_body will be executed again, When the loop_condition becomes false, the program control passes to the statement after the loop_body.

Nested Loops :-Any looping construct can also be nested within any other looping construct . Let us look at the following example showing the nesting of a for() loop within the scope of another for() loop :

```
for(int i = 1 ; i<=2 ; i++)  Outer for( ) loop
{
    for( int j = 1 ; j<=3 ; j++)  Inner for( ) loop
    {
        cout<< i * j <<endl ;
    }
}
```

For each iteration of the outer for loop the inner for loop will iterate fully up to the last value of inner loop iterator. The situation can be understood more clearly as :

1st Outer Iteration
i= 1

1st Inner Iteration
 j = 1 , output : 1 * 1 = 1
 2nd Inner Iteration
 j = 2 , output : 1 * 2 = 2
 3rd Inner Iteration
 j = 3 , output : 1 * 3 = 3

2nd Outer Iteration
 i= 2

1st Inner Iteration
 j = 1 , output : 2 * 1 = 1
 2nd Inner Iteration
 j = 2 , output : 2 * 2 = 4
 3rd Inner Iteration
 j = 3 , output : 2 * 3 = 6

You can observe that j is iterated from 1 to 2 every time i is iterated once.

Jump Statements:-These statements unconditionally transfer control within function . In C++ four statements perform an unconditional branch :

1. return
2. goto
3. break
4. continue

1. **return Statement:-** The return statement is used to return from a function. It is useful in two ways:

- (i) An immediate exit from the function and the control passes back to the operating system which is main's caller.
- (ii) It is used to return a value to the calling code.

2. **goto statement :-** A goto Statement can transfer the program control anywhere in the program. The target destination of a goto statement is marked by a *label*. The target label and goto must appear in the same function. The syntax of goto statement is:

```
goto label;
```

```
:
```

```
label :
```

Example :

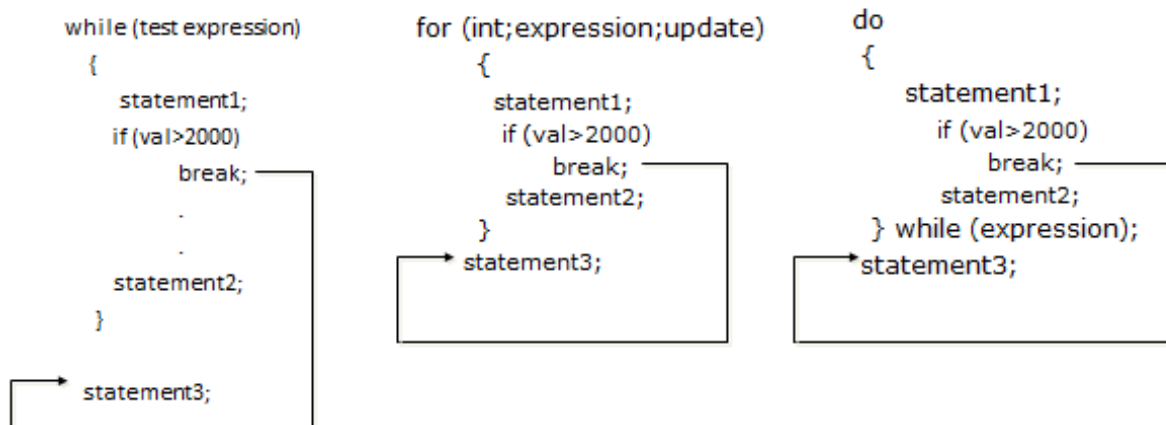
```
a= 0;
```

```
start :
```

```
    cout<<"\n" <<++a;
```

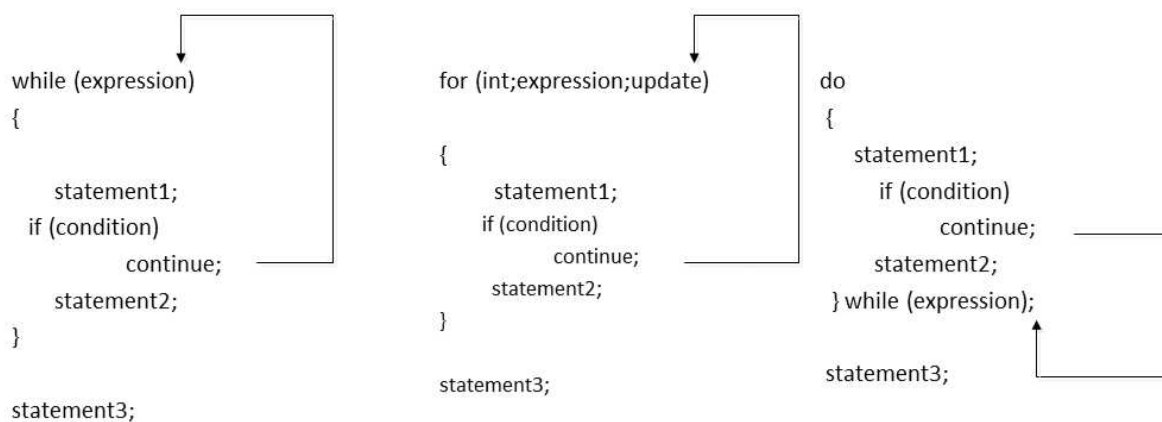
```
    if(a<50) goto start;
```

3. **break Statement :-** The break statement enables a program to skip over part of the code. A break statement terminates the smallest enclosing while, do-while, for or switch statement. Execution resumes at the statement immediately following the body of the terminated statement. The following figure explains the working of break statement:



The Working of a Break Statement

4. **continue Statement**:- The continue is another jump statement like the break statement as both the statements skip over a part of the code. But the continue statement is somewhat different from break. Instead of forcing termination, it forces the next iteration of the loop to take place, skipping any code in between. The following figure explains the working of continue statement:



The Working of Continue Statement

Functions :- Function is a named group of programming statements which perform a specific task and return a value.

There are two types of functions:-

1. Built-in (Library) functions
2. User defined functions

Built-in Functions (Library Functions) :- The functions, which are already defined in C++ Library (in any header files) and a user can directly use these function without giving their definition is known as built-in or library functions. e.g., sqrt(), toupper(), isdigit() etc.

Following are some important Header files and useful functions within them :

stdio.h (standard I/O function)	gets(), puts()
cctype.h (character type function)	isalnum(), isalpha(), isdigit(), islower(), isupper(), tolower(), toupper()
string.h (string related function)	strcpy(), strcat(), strlen(), strcmp(), strncmpi(), strrev(),strupr(), strlwr()
math.h (mathematical function)	fabs(), pow(), sqrt(), sin(), cos(), abs()
stdlib.h	randomize(), random()

The above list is just few of the header files and functions available under them , but actually there are many more. The calling of library function is just like User defined function , with just few differences as follows:

- i) We don't have to declare and define library function.
- ii) We must include the appropriate header files , which the function belongs to, in global area so as these functions could be linked with the program and called.

Library functions also may or may not return values. If it is returning some values then the value should be assigned to appropriate variable with valid datatype.

gets() and puts() : these functions are used to input and output strings on the console during program run-time.

gets() accept a string input from user to be stored in a character array.
puts() displays a string output to user stored in a character array.

isalnum() , isalpha() , isdigit() : checks whether the character which is passed as parameter to them are alphanumeric or alphabetic or a digit ('0' to '9') . If checking is true functions returns 1.

islower() , isupper() , tolower() , toupper() : islower() checks whether a character is lower case , isupper() check whether a character is upper case . tolower() converts any character passed to it in its lower case and the toupper() convert into upper case.

fabs() , pow() , sqrt() , sin() , cos() , abs() :

randomize() , random() :

The above functions belongs to header file stdlib.h . Let us observe the use of these functions :

randomize() : This function provides the seed value and an algorithm to help random() function in generating random numbers. The seed value may be taken from current system's time.

random(<int>) : This function accepts an integer parameter say x and then generates a random value between 0 to x-1

for example : random(7) will generate numbers between 0 to 6.

To generate random numbers between a lower and upper limit we can use following formula

$$\text{random}(U - L + 1) + L$$

where U and L are the Upper limit and Lower limit values between which we want to find out

random values.

For example : If we want to find random numbers between 10 to 100 then we have to write code as :

```
random(100 -10 +1) + 10 ; // generates random number between 10 to 100
```

User-defined function :- The functions which are defined by user for a specific purpose is known as user-defined function. For using a user-defined function it is required, first define it and then using.

Function Prototype :- Each user define function needs to be declared before its usage in the program. This declaration is called as function prototype or function declaration. Function prototype is a declaration statement in the program and is of the following form :

```
Return_type function_name(List of formal parameters) ;
```

Declaration of user-defined Function: In C++ , a function must be defined, the general form of a function definition is :

```
Return_type function_name(List of formal parameters)
{
    Body of the function
}
```

Where Return_type is the data type of value return by the function. If the function does not return any value then void keyword is used as return_type.

List of formal parameters is a list of arguments to be passed to the function. Arguments have data type followed by identifier. Commas are used to separate different arguments in this list. A function may be without any parameters, in which case , the parameter list is empty.

statements is the function's body. It is a block of statements surrounded by braces { }.

Function_name is the identifier by which it will be possible to call the function.

e.g.,

```
int addition (int a, int b)
{
    int r ;
    r=a+b ;
    return (r) ;
}
```

Calling a Function:- When a function is called then a list of actual parameters is supplied that should match with formal parameter list in number, type and order of arguments.

Syntax for calling a function is:

```
function_name ( list of actual parameters );
```

e.g.,

```
#include <iostream>
int addition (int a, int b)
{ int r;
  r=a+b;
  return (r); }
void main ( )
{ int z ;
  z = addition (5,3);
  cout<< "The result is " << z;
}
```

The result is 8

```
int addition (int a, int b)
```



```
z = addition ( 5 , 3 );
```

```
int addition (int a, int b)
```



```
z = addition ( 5 , 3 );
```


Call by Value (Passing by value) :- The call by value method of passing arguments to a function copies the value of actual parameters into the formal parameters, that is, the function creates its own copy of argument values and then use them, hence any change made in the parameters in function will not reflect on actual parameters. The above given program is an example of call by value.

Call by Reference (Passing by Reference) :- The call by reference method uses a different mechanism. In place of passing value to the function being called, a reference to the original variable is passed. This means that in call by reference method, the called function does not create its own copy of original values, rather, it refers to the original values only by different names i.e., reference. Thus the called function works the original data and any changes are reflected to the original values.

// passing parameters by reference

```

#include <iostream.h>
void duplicate (int& a, int& b, int& c)
{
    a*=2;
    b*=2;
    c*=2;
}

void main ()
{
    int x=1, y=3, z=7;
    duplicate (x, y, z);
    cout <<"x="<< x <<" , y="<< y <<" , z="<< z;
}

```

```

void duplicate (int& a, int& b, int& c)
                ↑x   ↑y   ↑z
duplicate ( x , y , z );

```

output :x=2, y=6, z=14

The ampersand (&) (address of) specifies that their corresponding arguments are to be passed *by reference* instead of *by value*.

Constant Arguments:-In C++ the value of constant argument cannot be changed by the function. To make an argument constant to a function, we can use the keyword `const` as shown below:

```
int myFunction( const int x , const int b );
```

The qualifier `const` tells the compiler that the function should not modify the argument. The compiler will generate an error when this condition is violated.

Default Arguments :- C++ allows us to assign default value(s) to a function's parameter(s) which is useful in case a matching argument is not passed in the function call statement. The default values are specified at the time of function definition. e.g.,

```
float interest ( float principal, int time, float rate = 0.70f)
```

Here if we call this function as:

```
si_int= interest(5600,4);
```

then `rate = 0.7` will be used in function.

Formal Parameters:- The parameters that appear in function definition are formal parameters.

Actual Parameters :- The parameters that appears in a function call statement are actual parameters.

Functions with no return type (The use of void):- If you remember the syntax of a function declaration:

```
Return_type function_name(List of formal parameters)
```

you will see that the declaration begins with a type, that is the type of the function itself (i.e., the data type of value that will be returned by the function with the return statement). But what if we want to return no value?

Imagine that we want to make a function just to show a message on the screen. We do not need it to return any value. In this case we should use the void type specifier for the function. This is a special specifier that indicates absence of type.

The return Statement :- The execution of return statement, it immediately exit from the function and control passes back to the calling function (or, in case of the main(), transfer control back to the operating system). The return statement also returns a value to the calling function. The syntax of return statement is:

```
return ( value);
```

Scope of Identifier :- The part of program in which an identifier can be accessed is known as scope of that identifier. There are four kinds of scopes in C++

- (i) Local Scope :- An identifier declare in a block ({ }) is local to that block and can be used only in it.
- (ii) Function Scope :- The identifier declare in the outermost block of a function have function scope.
- (iii) File Scope (Global Scope) :- An identifier has file scope or global scope if it is declared outside all blocks i.e., it can be used in all blocks and functions.
- (iv) Class Scope :- A name of the class member has class scope and is local to its class.

Lifetime :- The time interval for which a particular identifier or data value lives in the memory is called Lifetime of the identifier or data value.

ARRAYS :-Array is a group of same data types variables which are referred by a common name.

An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.

Need for Arrays:- Suppose we have to store roll number & marks of 50 students in a program , then in all we need 100 variable each having a different name. Now remembering and managing these 100 variables is not easy task and it will make the program a complex and not-understandable program.

But if we declare two array each having fifty elements one for roll numbers and another for marks. Now, we have only two names to remember. Elements of these array will be referred to as Arrayname[n], where n is the element number in in the array . Therefore arrays are very much useful in a case where quit many elements of the same data types need to be stored and processed.

The element number s in parenthesis are called subscripts or index. The subscript, or index of an element designates its position in the array's ordering.

Types of Array :-Arrays are of two types:

1. One-dimensional Arrays

2. Multi-dimensional Arrays (But we have to discussing only Two-dimensional arrays in our syllabus)

One –dimensional Arrays :- The simplest form of an array is one - dimensional. The array itself is given a name and its elements are referred to by their subscripts. In C++ , an array must be defined before it can be used to store information. The general form of an array declaration is:

```
Data_type Array_name[size];
```

For example the following statement declares an array marks of int data type , which have 50 elements marks[0] to marks[49].

```
int marks[50] ;
```

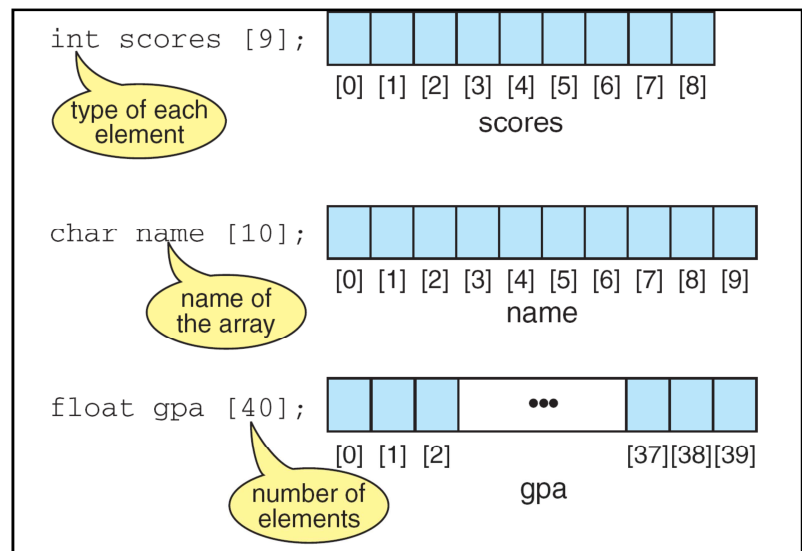
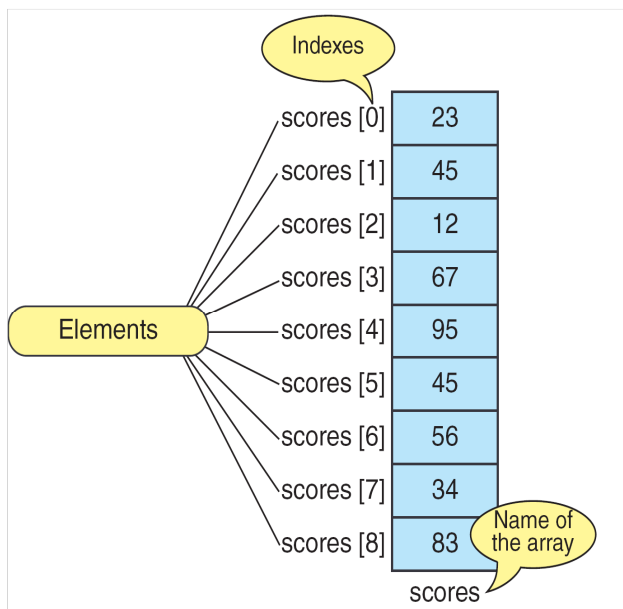
Referencing Array Elements:- We can reference array elements by using the array’s name & subscript (or index). The first element has a subscript of 0. The last element in an array of length *n* has a subscript of *n-1*.

If we declare an array:

```
int scores[9];
```

then its 10 element will be referred as:

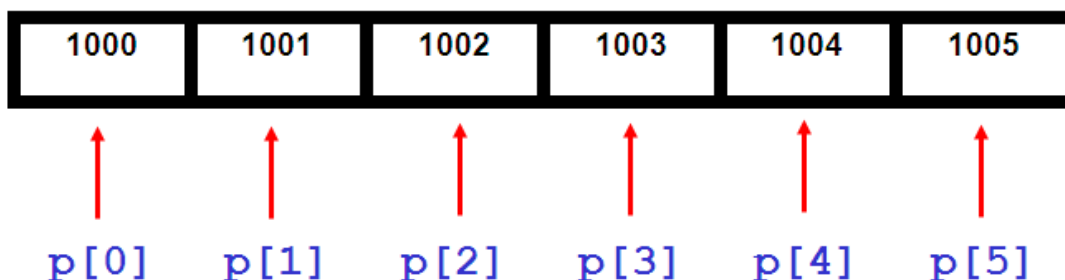
scores[0], scores[1], , scores[8]



Memory Representation of Single Dimensional Arrays:-Elements of single dimensional array are stored in contiguous memory location in their index order. For example , an array p of type char with 6 elements declared as:

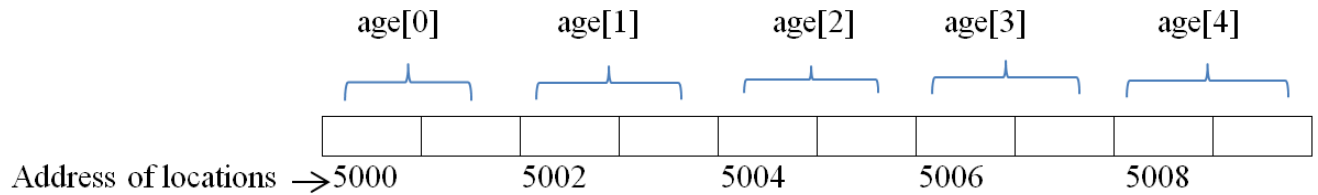
```
char p[6];
```

will be represented in memory as shown below:

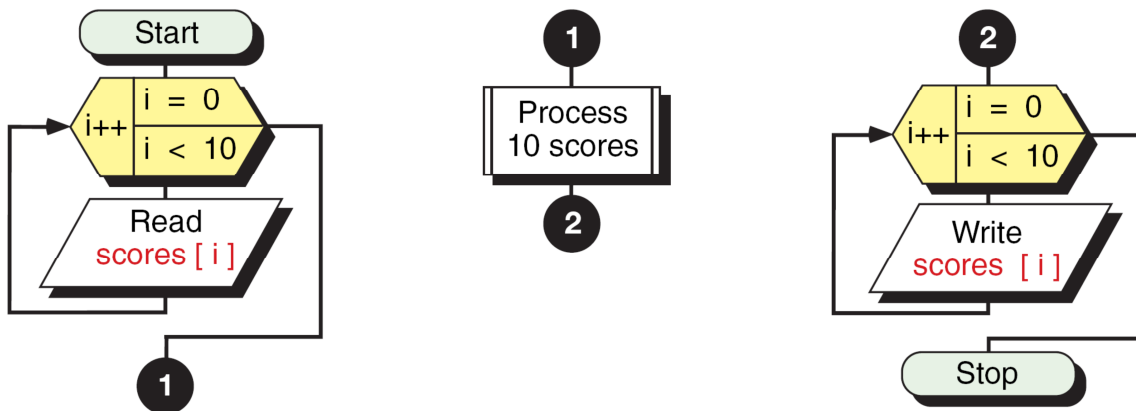


If we declare an int array age with 5 elements as:

```
int age[5];
```



Accepting Data in Array from User (Inputting Array elements) and Printing Array elements:-
 Since we can refer to individual array elements using numbered indexes, it is very common for programmers to use **for** (or any) loops when processing arrays.



<p>General form of for loop for Reading elements of array (1-D)</p> <pre>for (int i=0; i< size; i++) { cout<<"Enter Array Element "<<i+1; cin>>Array_Name[i]; }</pre>	<p>Generally processing part may be include with in the loop of reading or printing, otherwise a same type separate loop may be used for processing</p>	<p>General form of for loop for printing elements of array (1-D)</p> <pre>for (int i=0; i< size; i++) { cout<<Array_Name[i]<< " , "; }</pre>
--	---	---

1. Initializing arrays.

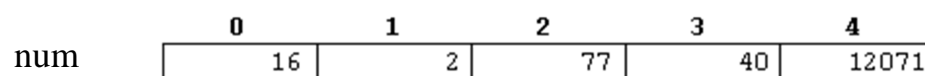
When we declare an array, we have the possibility to assign initial values to each one of its elements by enclosing the values in braces { }. Syntax for initialization of one-dimensional array is:

```
Data_type Array_name[sise] = { value list } ;
```

For example:

```
int num [5] = { 16, 2, 77, 40, 12071 };
```

This declaration would have created an array like this:



The amount of values between braces { } must not be larger than the number of elements that we declare for the array between square brackets [].

C++ allows us to skip the size of the array in an array initialization statement (by leaving the square brackets empty []). In this case, the compiler will assume a size for the array equal to the number of values given between braces { }:

```
int num [ ] = { 16, 2, 77, 40, 12071 };
```

After this declaration, the size of array num will be 5, since we have provided 5 initialization values.

String as an array:- C++ does not have a string data type rather it implements string as single dimensional character array. A string is defined as a character array terminated by a null character '\0'. Hence to declare an array strg that holds a 10 character string, you would write :

```
char strg[11];
```

Two dimensional array : A two dimensional array is a continuous memory location holding similar type of data arranged in row and column format (like a matrix structure).

Declaration of 2-D array:- The general syntax used for 2-D array declaration is:

```
Data_type Array_name [R][C] ;
```

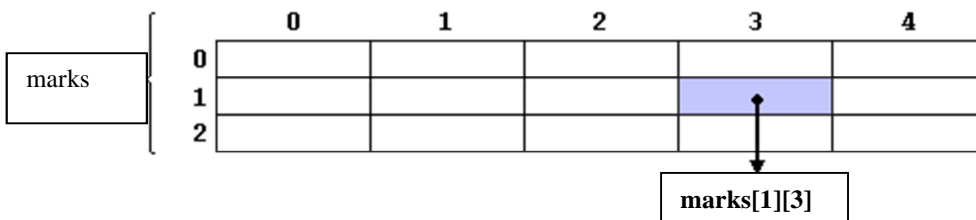
Where R represent number of rows and C represent number of columns in array.

For example,

```
int marks [3][5];
```

and, for example, the way to reference the second element vertically and fourth horizontally in an expression would be:

```
marks[1][3]
```



Initialization of 2-D Array:- Two dimensional arrays are also initialize in the same way as single dimensional array . e. g.,

```
int cube[5][2] = { 1,1, 2,8,3,27,4,64,4,125 };
```

will initialize cube[0][0]=1, cube[0][1]=1, cube[1][0]=2, cube[1][1]=8 , cube[2][0]=3 and so on.

2-D array can also be initialized in unsized manner. However the first index can be skiped , the second index must be given. e.g.,

```
int cube[ ][2] = { 1,1, 2,8,3,27,4,64,4,125 };
```

is also valid.

Accepting Data in 2-D Array from User (Inputting 2-D Array elements) and Printing 2-D Array elements:- Since We must reference both, a row number and a column number to input or output values in a two-dimensional array. So, we use a nested loop (generally nested for loop) , when processing 2-D arrays.

General form of for loop for Reading elements of 2-D array	Generally processing part may be include within the loop of reading or printing, otherwise a same type separate nested loop may be used for processing	General form of for loop for printing elements of 2-D array
<pre>for (int i=0; i< R; i++) { cout<<"Enter Row "<<i+1; for (int j=0; j<C ; j++) cin>>Array_Name[i][j]; }</pre>		<pre>for (int i=0; i< R; i++) { for (int j=0; j<C ; j++) cout<<Array_Name[i][j] <<'t'; cout<<'n'; }</pre>

Where R represent number of rows and C represent number of columns in array.

Array of Strings:- An array of string is a two-dimensional character array. The size of first index determines the number of strings and size of second index determines maximum length of each string. The following statement declares an array of 10 strings , each of which can hold maximum 50 valid characters.

```
char string[10][51] ;
```

Notice that the second index has been given 51, because 1 extra size is given for store null character '\0'.

User Defined Data Types:-

The C++ language allows you to create and use data types other than the fundamental data types. These types are called user-defined data types.

Structures:- In C++, a structure is a collection of variables that are referenced by a single name. The variable can be of different data types. They provide a convenient means of keeping related information together.

Defining Structure :-

A Structure is defined by using the Keyword struct. The General Syntax for defining a Structure is :

Syntax :

```
struct< Name of Structure >
{
  <datatype>< data-member 1>;
  <datatype>< data-member 2>;
  <datatype>< data-member 3>;
  ...
  ...
  <datatype>< data-member n>;
};
```

Example :

A Proper example following the previous syntax could be :

```
struct WORKER
{
  char name[45];
  char gender ;
  int age ;
}
```

Structure name

data-member

```
float rate ;
```

```
};
```

Declaring Structure Variable :-

This is similar to variable declaration. We can declare the variable of structure type using two different ways either with structure definition or after structure definition.

The following syntax shows the declaration of structure type variables with structure definition:

```
struct< Name of Structure >
{
    <datatype>< data-member 1>;
    <datatype>< data-member 2>;
    <datatype>< data-member 3>;
    ...
    ...
    <datatype>< data-member n>;
} var1, var2,....., varn ;
```

We can declare the structure type variables separately (after defining of structure) using following syntax:

```
Structure_name var1, var2, ..... ....., var_n;
```

Accessing Structure Elements :- To access structure element , dot operator is used. It is denoted by (.). The general form of accessing structure element is :

```
Structure_Variable_Name.element_name
```

Initializing of Structure elements:- You can initialize members of a structure in two ways. You can initialize members when you declare a structure or you can initialize a structure with in the body of program. For Example:

First Method:-

```
#include <iostream.h>
#include <conio.h>
void main()
{
    // Declaring structure at here
    struct Employee
    {
        int empcode;
        float empsalary;
    };
    Employee emp1 = {100, 8980.00} ;           // emp1 is the structure variable ,
                                              // which is also initialize with declaration

    clrscr();
    int i;           // declares a temporary variable for print a line
                   // Printing the structure variable emp1 information to the screen
    cout<< "Here is the employee information : \n";
    for (i = 1; i <= 32; i++)
        cout<< "=";
    cout<< "\nEmployee code : " << emp1.empcode;
    cout<< "\nEmployee salary : " << emp1.empsalary;
}
```

Second Method:-

```
#include <iostream.h>
#include <conio.h>
void main()
{
    // Declaring structure here
    struct Employee
    {
        int empcode;
        float empsalary;
    } emp1;      // emp1 is the structure variable
    clrscr();
    int i;      // declares a temporary variable for print a line
    // Initialize members here
    emp1.empcode = 100;
    emp1.empsalary = 8980.00;
    // Printing the structure variable emp1 information to the screen
    cout<< "Here is the employee information : \n";
    for (i = 1; i <= 32; i++)
        cout<< "=";
    cout<< "\nEmployee code : " << emp1.empcode;
    cout<< "\nEmployee salary : " << emp1.empsalary;
}
```

Structure variable assignments :-

We know that every variable in C++ can be assigned any other variable of same data type i.e :

```
if int a = 7 ; b = 3;
```

we can write :

```
a = b ; // assigning the value of b to variable a
```

Similar is the case with Structure variables also , i.e :

```
if WORKER w1 = {"Ramu" , 'M' , 17 , 100.00};
```

```
WORKER w2;
```

we can write :

```
w2 = w1 ; // assigning the corresponding individual // data member values of w1 to
```

```
Worker w2;
```

or

```
WORKER w2 = w1;
```

Note : Both structure variables must be of same type (i.e WORKER in this case)

There is a member wise copying of member-wise copying from one structure variable into other variable when we are using assignment operator between them.

So, it is concluded that :

```
Writing : strcpy (w1. name,w2.name) ;
```

```
w1.gender = w2.gender ;
```

```
w1.age = w2.age ;
```

```
w1.rate = w2.rate ;
```

is same as :

```
w1 = w2 ;
```


Array of Structure :- We can define an array of structure by using following syntax :

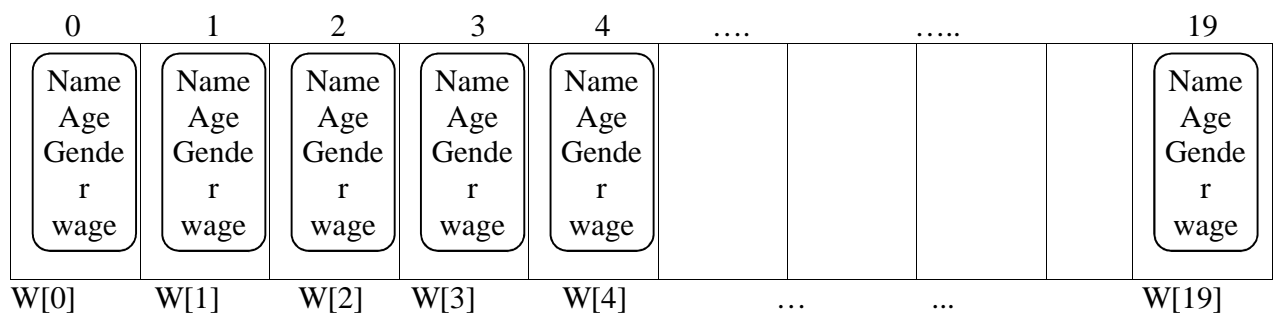
```
<structure_name><array_name>[ size ];
```

where :
 structure_name is the name of the structure which you have created.
 array_name is any valid identifier
 size is a positive integer constant.

Example : to create array of 20 Workers we can have :

```
Worker W[20];
```

The above structure could be visualized as :



Each of the elements of the array is itself a structure hence each of them have all the four components.

Function with Structure variable:-

We can also pass a structure variable as function parameter / arguments. The benefit is that the structure carries a bundled information to the structure. The prototype of such a function would be

```
<return_type> function_name(<structure_name><var> , ... , ... );
```

Let us understand the concept with following function ,which increases the wage of a female worker by passed percent

```
void increaseWage( Worker & w , float incr )
{
    if( w.gender == 'F' || w.gender == 'f' )
    {
        w.wage += w.wage* (incr /100) ;
    }
}
```

Look at the highlighted parameter, we have passed formal structure variable as reference, so that the increment is reflected back in actual parameter.

Similarly, if we don't want to pass our structure variable as reference we can do so , but then we have to return the modified structure variable back. This can be achieved in above function, if we

take return type as structure name. Let us modify the function so that it returns a structure :

```
Worker increaseWage( Worker w , float incr)
{
    if( w.gender == 'F' || w.gender == 'f' )
    {
        w.wage + = w.wage* (incr /100) ;
    }
    return w ;
}
```

Nested structure :- A Structure within another structure is known as nested structure. A structure containing an another structure as valid element is known as nested structure.

e.g.

```
struct address
{
int houseno;
char city[20];
char area[20];
long int pincode;
};
```

```
struct employee
{
int empno;
char name[30];
char design[20];
char department[20];
address ad; // nested structure
}
```

Declaration:

```
employee e;
```

Input /Output :

```
cin>>e.ad.houseno;           // members are accessed using dot(.) operator.
cout<<e.ad.houseno;
```

typedef :- The typedef keyword allows to create a new names for data types. the syntax is:

```
typedef existing_data_type new_name ;
```

e.g.,

```
typedef int num;
```

#define Preprocessor Directive :-The # define directive create symbolic constant, constants that are represent as symbols and macros (operation define as symbols). The syntax is:

```
#define identifier replacement_text ;
```

When this line appears in a file, all subsequent occurances of identifier in that file will be replaced by replacement_text automatically before the program is compiled. e.g.,

Program: Program to calculate area of a circle by using #define preprocessor directive.

```
#include <iostream.h>
```

```
#define PI 3.14159
```

```
#define CIRCLE_AREA(x) (PI * (x) * (x))
```

```
void main()
{
    float area;
    int r;
    cout<< "Enter the radius : ";
    cin>> r;
    area = CIRCLE_AREA(r);
    cout<< "Area is : " << area;
}
```

Object Oriented Programming

Classes and Objects:-

Class:-

A class is a user defined data that contained data as well as function together.

or

A class is a way to bind the data and function together in a single unit.

or

A class is a group of objects that share common properties and relationships

Note:-

1. The variables and functions enclosed in a class are called data member and member functions.
2. The variable of class are called objects or instance of a class .
3. Classes are basic user defined data type of object oriented programming language
4. The difference between a class and structure is that ,by default ,all the members of a class are private while by default ,all the members of structure are public

Class Declaration:-

```
Class <classname>
{
    private:
        data members;
        member functions;
    protected:
        data members;
        member functions;
    public:
        data members ;
        member functions:
};
```

Note:- The data members and member functions of class are grouped under three sections.

1. private
2. protected
3. public

Explanation of private ,protected & public access modifier:-

private :-

- By private we mean that members can be accessed only from within the class i.e member data can be accessed by the member function .
- The private data members are not accessible to the outside works (i.e outside the class)
- By default members of a class are private.
- Private members are not inheritable.

Protected:-

- The members which are declared as protected ,can only be accessed by member functions and friends function function of that class.

- Protected members are similar to private members ,the only difference is that protected members are inheritable .

Public:-

The members which are declared as public can be accessed from out side classes also.

Note:- By default ,the members of a class are private .if both the lables ,are missing,then by difault ,all the members are private .Such a class is completely hidden from outside worls.

Example:-

```
class Student
{
    int rollno;
    float marks;
public:
    void getdata(int x,flot y);
    void display(void);
};
```

Array of Objects:- An array of variables that are of the type Class .such variables are called array of objects.

Example.

```
class employee
{
    char name[50];
    int age;
public:
    void getdata(void);
    void showdata(void);
};
```

The employee is a user defined data type and can be used to create objects that relate to different categories of employees.

```
employee    manager[4];
employee    engineer[15];
employee    workers[200];
```

Creating Objects:-

1. An object is an instance of class .
2. In general a class is a user defined data type .while an object is an instance of class
3. Once a class has been declared ,we can create variables of that type by using class name.
4. An object occupies memory space.

Let Student be the name of class

Student S1, S2;

or

Class Student

{

} S1,S2;

Accessing Class Members:- After creating the objects there is a need to access the class members .This can be done using (.) operator .

Syntax:

```
Objectname . Datamember  
Objectname.memberfunction
```

As:

```
S1.getdata(129,100)  
s1.display()  
s1.rollno=129 // illegal statement
```

Note:- A variable declared public can be accessed by the objects directly.

//Example for public data members

```
#include<iostream.h>  
#include<conio.h>  
void main()  
{  
    clrscr();  
    class date  
    {  
        public:  
            int day;  
            int month;  
            int year;  
    };  
  
    date d1;  
    d1.day=26;  
    d1.month=6;  
    d1.year=2012  
    cout<<" Date is"<<d1.day<<"/" <<d1.months<<"/" <<d1.year<<endl;  
    getch();  
}
```

Example 2:-Write a program to make use of simple arithmetic operations such as addition,subtraction,multiplication division using the concept class.

```
#include <iostream.h>  
#include<conio.h>  
class calculator  
{  
    private :  
        int x;  
        int y;  
    public :  
        void getdata();  
        {  
            cout<<" Enter the two numbers:" << endl;  
            cin>> x >> y;  
        }  
        void display()
```

```

        {
            cout<<"x="<<x<<endl;
            cout<<"y="<<y<<endl;
        }
void sum()
{
cout<<" addition of x & y ="<<x+y<<endl;
}
void diff()
{
cout << " diff of x & y =" <<x-y<<endl;
}
void mul()
{
cout<<" mul of x & y ="<<x*y<<endl;
}
void div()
{
cout<< " div of x & y="<< x/y<<endl;
}
};
Void main()
{
    clrscr();
    calculator C1;
    c1.getdata();
    c1.display();
    c1.sum();
    c1.diff();
    c1.mul();
    c1.div();
    getch();
}

```

Class method definition:- The data members of a class must be declared within the body of the class ,whereas member functions of the class can be defined in following ways:

- 1.Out side the class definition
2. Inside the class definition

Member function outside the class definition:-

Note:- 1. declare function prototype within the body of a class and then define it outside the body of class.

2. By using scope resolution operator (::) we bind the function to the class.

Syntax:-

```

Return type classname :: functionname(argument declaration)
{
    function body
}

```

Example:

```

#include<iostream.h>
#include<conio.h>
class product
{
    int qty;
    float cost;
    Public:
        void getdata(int a, float b);
        void putdata(void);
};
void product :: getdata(int a,float b)
{
    qty= a;
    cost=b;
}
void product :: putdata(void)
{
    cout<<" qty="<< qty<<endl;
    cout<<"cost=" << cost<<endl;
}
void main()
{
    clrscr();
    product p;
    cout<<" The object ps is "<< endl;
    p.getdata(75,255.60);
    p.putdata();
    product Q;
    cout<<" The object Q is =" << endl;
    Q.getdata(100,125.25);
    Q.putdata();
    getch();
}

```

Note:- label product:: informs the compiler that the functions getdata & putdata are the

2. Member function inside the class definition.

Note:- When a function is defined inside a class ,it is treated as inline function.normally small functions are defined inside the class definition.

Example:-

```

#include<iostream.h>
#include< conio.h>
class date
{
    private:
        int day;
        int month;
        int year;
}

```



```

        public:
            void getdata(int x,int y,int z)
            {
                day=x;
                month=y;
                year=z;
            }
            void display()
            {
                cout<< day<<month<<year <<endl;
            }
        }
    }
    Void main()
{
    clrscr();
    date d1,d2,d3;
    d1.getdata(30,12,2012);
    d2.getdata(23,11,2011);
    d3.getdata(24,05,2010);
    cout<< " you have entered following date";
    d1.display();
    d2.display();
    d3.display();
    getch();
}

```

Array of Objects:- An array of variables that are of the type Class .Such variables are called array of objects.

Example.

```

class employee
{
    char name[50];
    int age;
    public:
        void getdata(void);
        void showdata(void);
};

```

The employee is a user defined data type and can be used to create objects that relate to different categories of employees.

```

employee    manager[4];
employee    engineer[15];
employee    workers[200];

```

Example:-

```

#include<iostream.h>
#include<conio.h>
class employee
{
    char name[50];

```

```

        int age;

        public:
            void getdata(void);
            void showdata(void);
};
void employee:: getdata(void)
{
    cout<<"enter name=";
    cin>>name;
    cout<<"enter age=";
    cin>>age;
}
void employee::showdata(void)
{
    cout<<name<<endl<<age;
}
void main()
{
    clrscr();
    int size=4;
    employee manager[size];
    for(int i=0; i<size;i++)
    {
        cout<<"Enter Details of managers";
        manager[i].getdata();
    }
    cout<<endl;
    for(i=0;i<size;i++)
    {
        cout<<"Details of Managers is=";
        manager[i].putdata();
    }
    getch();
}

```

Friend Function:- A friend function is not a member function ,has full access rights to the private members of the class.

Syntax :-

```

class ABC
{
    -----
    -----
    public:
    -----
    -----
    friend void xyz(void);
}

```

Characteristics of friend function:

- It is not in the scope of the class to which it has been declared as friend.
- Since it is not in the scope of the class, it cannot be called using the object of that class.
- It can be invoked like a normal function without the help of any object.
- Unlike member functions, it cannot access the member names directly and has to use an object name and dot membership operator with each member name.
- It can be declared either in the public or the private part of a class without affecting its meaning.

Usually, it has the objects as arguments.

Example:-

```
#include<iostream.h>
#include<conio.h>
Class sample
{
    int a;
    int b;
    public:
        void setvalue()
        {
            a=25;
            b=40;
        }
    friend float mean(sample s);
};
float mean (sample s)
{
    return float (s.a+s.b)/2;
}
void main()
{
    sample x;
    x.setvalue();
    cout<<mean(x);
}
// A function friendly to two classes
#include<iostream.h>
class XYZ
{
    int x;
    public:
        void setvalue(int i)
        {
            x=i;
        }
    friend void max(XYZ,ABC);
};
class ABC
{
```

```

int a;
public:
    void setvalue(int i)
    {
        a=i;
    }
    friend void max(XYZ,ABC);
};

```

```

void max(XYZ m, ABC n)
{
    if (m.x>=n.a)
        cout<<m.x;
    else
        cout<<n.x;
}

```

```

void main()
{
    ABC abc;
    abc.setvalue(10);
    XYZ xyz;
    xyz.setvalue(20);
    max(xyz,abc);
}

```

Object as function Arguments: -An object can be passed as an argument to functions in two ways.

1. **pass by value**

2. **pass by reference.**

// Example of call by value

```
#include<iostream.h>
```

```
#include<conio.h>
```

```
class class_2;
```

```
class class_1
```

```
{
```

```
int value1;
```

```
public:
```

```
void indata(int a)
```

```
{
```

```
value1=a;
```

```
}
```

```
void display(void)
```

```
{
```

```
cout<<value1<<"\n";
```

```
friend void exchange(class _1 obj1, class_2 obj2);
```

```
}
```

```
class class2
```

```
{
```

```
int value2;
```

```

public:
    void indata(int a)
    {
        value2=a;
    }
    void display(void )
    {
        cout<<value2<<“\n”;
    }
    friend void exchange (class_1 t1, class_2 t2);
};
void exchange(class_1 T1,class_2 T2)
{
    int temp;
    temp=T1.value1;
    T1.value1=T2.value2;
    T2.value2=temp;
}
void main()
{
    class_1 C1;
    class_2 C2;
    C1.indata(100);
    C2.indata(200);
    cout<<“ values before exchange “<<“\n”;
    C1.display();
    C2.display();
    exchange(C1,C2);
    cout<<“ values after exchange”<<“\n”;
    C1.display();
    C2.display();
}

```

Object as function Arguments:-An object can be passed as an argument to functions in two ways.

1. pass by value
2. pass by reference.

// Example of call by value

```

#include<iostream.h>
#include<conio.h>
class class_2;
class class_1
{
    int value1;
public:
    void indata(int a)
    {
        value1=a;
    }
    void display(void)

```

```

        {
            cout<<value1<<“\n”;
            friend void exchange(class _1 obj1, class_2 obj2);
        }

class class2
{
    int value2;
public:
    void indata(int a)
    {
        value2=a;
    }
    void display(void )
    {
        cout<<value2<<“\n”;
    }
    friend void exchange (class_1 t1, class_2 t2);
};
void exchange(class_1 &T1,class_2 &T2)
{
    int temp;
    temp=T1.value1;
    T1.value1=T2.value2;
    T2.value2=temp;
}
void main()
{
    class_1 C1;
    class_2 C2;
    C1.indata(100);
    C2.indata(200);
    cout<<“ values before exchange “<<“\n”;
    C1.display();
    C2.display();
    exchange(C1,C2);
    cout<<“ values after exchange”<<“\n”;
    C1.display();          C2.display();          }
}

```

Constuctor and Destructor

Constructor:-

A constructor is a special member function which is used to initialize the object of a class.

Note:-

- A constructor is distinct from other member functions because its name is same as the name of the class .It is executed automatically when an object is declared .The only restriction that applies to constructor is that it must not have a return type not even void.
- Constructor of a class is the first member function to be executed automatically when an object of the class is created.

The general syntax:

```
class classname
{
    -----
    -----
    -----
    classname( parameter list);
    -----
};
class name::classname(parameter list)
{
    -----
}
```

Example:

```
class Bank
{
    private:
        int x,y;
        -----
        -----
    public:
        Bank( );
};
Bank :: Bank()
{
    x=0;
    y=0;
}
void main()
{
    Bank B;
}
```

When an object B is created ,constructor will be called automatically and it will initialize the data members x and y=0

Example:-

```
#include<iostream.h>
#include<conio.h>
```

```

class fibonacci
{
    private:
        int fo,f1,fib;
    public:
        fibonacci();
        void increment();
        void display();
};

fibonacci::fibonacci()
{
    fo=0;
    f1=1;
    fib=fo+f1;
}
void fibonacci::increment()
{
    fo=f1;
    f1=fib;
    fib=fo+f1; }
void fibonacci:: display()
{
    cout<< fib<<“\t”;
}
void main()
{
    clrscr();
    fibonacci F;
    for(int i=0;i<=15;++i)
    {
        F.display();
        F.increment();
        getch();
    }
}

```

Default Constructor:

A constructor that accepts no parameters is called **default constructor**.

Syntax:

```

class First
{
    private:
    -----
    -----
    public:
    first();
    -----
}

```



```

};
-----
First :: First()
{
}

#include<iostream.h>
#include<conio.h>
class student
{
private:
char name[50];
int roll_no;
char sex;
int age;
float marks;
public:
student();
void putdata();
};
student:: student()
{
gets(name,"Donald");
roll_no=12;
sex='M';
age=34;
marks=90;
}
void student:: putdata()
{
cout<<name<<endl;
cout<<roll_no<<endl;
cout<<sex<<endl;
cout<<age<<endl;
cout<<marks<<endl;
}
void main()
{
clrscr();
student S;
S.putdata();
getch();
}

```

Parameterized constructor:-

The constructor that can take arguments are called Parameterized constructor.

Example:

```

class Bank
{
    private:
        int x,y;
    public:
        Bank(int a, int b)
        -----
        -----
};
Bank:: Bank(int a, int b)
{
    x=a;
    y=b;
}

```

In the parameterized constructor we pass the initial values as arguments to the constructor in two ways:

Implicit call

Explicit call

Implicit call:

```
Bank b1(5,11);
```

Explicit Call

```
Bank B1=Bank(5,11);
```

Example:-

```

#include<iostream.h>
#include<conio.h>
class Bank
{
    int x,y;
    public:
        Bank(int ,int);
        void putdata()
        {
            cout<<x<<“\n”<<y;
        }
};

```

```
Bank :: Bank(int a,int b);
```

```

{
    x=a;
    y=b;
}
void main()
{
    clrscr();
    Bank B1(5,11);
    Bank B2=Bank(6,34);
    cout<<“ object 1”;
    B1.putdata();
    cout<<“object 2”;
}

```

```

    B2.putdata();
    getch();
}

```

Copy Constructor:- A copy constructor takes a reference to its own class as parameter i.e a constructor having a reference to an instance of its own class as an argument is known as copy constructor.

Copy constructor are used in following situations:

1. The initialization of an object by another object of the same class.

Example:

```

    Sample S1; //default constructor

```

```

    Sample S2(S1);

```

or

```

    Sample S2=S1;

```

The general syntax

```

    classname ::classname(classname &)
#include<iostream.h>
class sample
{
    int x;
    public:
        sample()
        {
            x=0;
        }
        sample(int a)
        {
            x=a;
        }
        sample (sample & S)
        {
            x=s.x
        }
        void display(void)
        {
            cout<<x;
        }
};

void main()
{
    clrscr();
    Sample P(45);
    Sample Q(P);
    Sample R=P;
    Sample T;
    T=P;
    cout<<" \n value of p:";
}

```

```

P.display();
cout<<“\n value of Q:”;
Q.display();
cout<<“ \n value of R”;
R.display();
cout<<“\nValue of T:”;
T.display();
getch();
}

```

Constructor with default Arguments:-

Example:

```

item (int x,int y=0); // declaration

```

```

item:: item(int x,int y)
{
    a=x;
    b=y;
}

```

when we make object of item

```

item P(11);

```

11 will be assign to x and 0 to y;

```

item Q(11,17);

```

11 to x and 17 to y;

Example:

```

class student
{
    private :
    char name[50];
    int age;
    char sex;
    float marks;
    public:
        student(); // Constructor
        ~ student(); // Destructor
        void getdata();
        void putdata();
};

```

Note:- Object Are Destroyed In The Reverse Order Of Creation. So destructor is invoked in the reverse order of constructure

Program:-

```

#include<iostream.h>
#include<conio.h>
class sample
{

```

```

public:
    sample()
    {
        cout<<" object is born\n";
    }
    ~sample ()
    {
        cout <<" object is dead"
    }
};

void main()
{
    clrscr();
    sample S;
    cout<<" Main terminated "<<endl;
    getch();
}

```

SOME MPORTANT QUESTIONS TO REFRESH THE CONCEPT

Q1. What is constructor?

Ans. A member function with the same name as its class is called constructor and it is used to initialize the objects of that class type with a legal initial value.

Q2. What is destructor?

Ans. A destructor is a member function having same name as its class but preceded by ~ sign and it deinitialises an object before it goes out of scope.

Q3. What are different types of constructors?

Ans. The different types of constructors are Default constructor, Parameterized constructor and Copy constructor.

Q4. What is default constructor?

Ans. A constructor that accepts no parameter is called the default constructor.

Q5. What is parameterized constructor?

Ans. A constructor that accepts parameters for its invocation is known as parameterized constructor.

Q6. What is copy constructor?

Ans. A copy constructor is a constructor that defines and initializes an object with another object. It takes the form classname(classname). For a copy constructor there must be a default constructor or a parameterized constructor.

Q7. Answer the following questions after going through the following class:

```

class Seminar
{
    int Time;
public:
    Seminar(); //Function 1

```

```

void Lecture() //Function 2
{ cout<<"Lectures in the seminar on"<<endl;}
Seminar(int); //Function 3
Seminar(Seminar &abc); //Function 4
~Seminar() //Function 5
{ cout<<"Vote of thanks"<<endl;}
};

```

(i) In Object Oriented Programming, what is Function 5 referred as and when does it get invoked/called?

Ans : Function 5 is referred as destructor and it is invoked as soon as the scope of the object gets over.

(ii) In Object Oriented Programming, which concept is illustrated by Function 1, Function 3 and Function 4 all together?

Ans : Constructor Overloading (Polymorphism)

(iii) Which category of constructor - Function 1 belongs to? Write an example illustrating the calls for Function 1.

Ans : Default Constructor. Example to invoke function 1 → Seminar S;

(iv) Which category of constructor - Function 3 belongs to? Write an example illustrating the calls for Function 3.

Ans : Parameterised Constructor. Example to invoke function 3 → Seminar A(8);

(v) Which category of constructor - Function 4 belongs to? Write an example illustrating the calls for Function 4.

Ans : Copy Constructor. Example to invoke function 4 → Seminar S2(S);

Or Seminar S2 = S;

(vi) Write an example illustrating the calls for Function 3 explicitly.

Ans : Seminar A = Seminar(8);

(vii) Write an example illustrating the calls for Function 4 explicitly.

Ans : Seminar S2 = Seminar(S);

(viii) Write the complete definition for Function 1 to initialize Time as 30.

```

Seminar :: Seminar()
{ Time = 30; }

```

(ix) Write the complete definition for Function 3 to initialize Time with Mytime as parameter to the Function 3.

```

Seminar :: Seminar(int Mytime)
{ Time = Mytime; }

```

(x) Write the complete definition for Function 4.

```

Seminar :: Seminar(Seminar &abc)
{ Time = abc.Time; }

```

Q8. Answer the following questions after going through the following class:

```

class Complex
{
    int x;
    int y;
public:
    Complex(); //Function 1
    void disp() //Function 2

```

```

        {cout<<"The Complex number is : "<<x<<" + "<<y<<"i"<<endl;}
        Complex(int, int); //Function 3
        Complex(Complex &abc); //Function 4
};

```

(i) Which category of constructor - Function 1 belongs to? Write an example illustrating the calls for Function 1.

Ans : Default Constructor. Example to invoke function 1 → Complex C;

(ii) Which category of constructor - Function 3 belongs to? Write an example illustrating the calls for Function 3.

Ans: Parameterised Constructor. Example to invoke function 3→Complex C(6,8);

(iii) Which category of constructor - Function 4 belongs to? Write an example illustrating the calls for Function 4.

Ans : Copy Constructor. Example to invoke function 4 → Complex C2(C);

Or Complex C2 = C;

(iv) Write an example illustrating the calls for Function 3 explicitly.

Ans : Complex C = Complex(6,8);

(v) Write an example illustrating the calls for Function 4 explicitly.

Ans : Complex C2 = Complex(C);

(vi) Write the complete definition for Function 1 to initialize x as 10 and y as 20.

Ans : Complex :: Complex ()

{ x = 10 ; y = 20; }

(vii) Write the complete definition for Function 3 to initialize the data members with p and q as parameters to the Function 3.

Ans : Complex :: Complex(int p, int q)

{ x = p; y = q; }

(viii) Write the complete definition for Function 4.

Ans : Complex :: Complex (Complex &abc)

{ x = abc.x;

y = abc.y;

}

Inheritance -Extending Classes

Inheritance:- Inheritance is the process of creating new classes called derived classes from existing ones. i.e base classes.

Different form of Inheritance:

1. **Single Inheritance**
2. Multiple Inheritance
3. Hierarchical Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance.

Derived and Base Class:- A derived class extends its features by inheriting the properties of another class, called base class and adding features of its own.

The general syntax of derived class is :-

Class derived_class_name : visibility mode base class name

Visibility of Class Members

Base class visibility	Private derivation	Protected derivation	Public derivation
Private	Not inherited	Not inherited	Not inherited
Protected	Private	Protected	Protected
Public	Private	Protected	public

Single Level Inheritance:- Single inheritance is the process of creating new class (derived class) from an base class .i.e it creates a copy of base class and can add refinements of its own .

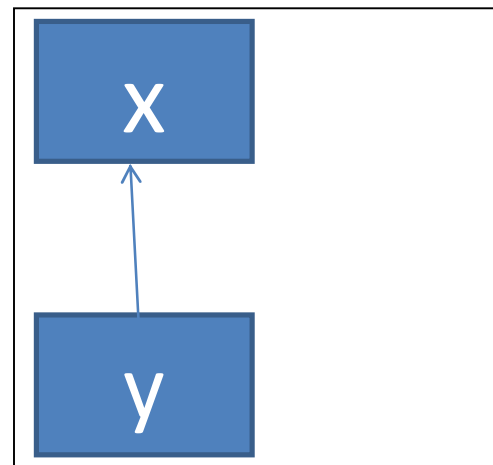
Public Derivation:-

Example:- #include <iostream.h>
#include<conio.h>


```

class item
{
    int x;
    public:
        int y;
        void getdata();
        int getvalue(void);
        void display(void);
};
class newitem : public item
{
    int z;
    public:
        void mul(void);
        void show(void);
};
void item :: getdata(void)
{
    x=9;
    y=17;
}
int item:: getvalue(void)
{
    return x;
}
void item:: display()
{
    cout<<"x="<<x<<endl;
}
void new item :: mul()
{
    z=y*getvalue();
}
void newitem:: show()
{
    cout<<"x="<<getvalue()<<endl;
    cout<<"y="<<y<<endl;
    cout<<"z=" <<z<<endl;
}
void main()
{
    clrscr();
    newitem n;
    n.getdata();
    n.mul();
    n.display();
    n.show();
    n.y=45;
    n.mul();
}

```



```

    n.show();
    getch();
}

```

Multilevel Inheritance:- Derivation of class from another derived class is called inheritance i.e when one derived class inherits from a class that itself inherits from another class is called multilevel inheritance.

Syntax:

```

Class X {.....}; // Base Class
Class Y: Public X; // Y derived from X
Class Z:Public Y; // Z derived from Y

```

```

#include<iostream.h>

```

```

#include<conio.h>

```

```

const int size=50;

```

```

class person

```

```

{

```

```

    char name[size];

```

```

    int age;

```

```

    char sex;

```

```

    public:

```

```

        void getdata()

```

```

        {

```

```

            cout<<" name="";

```

```

            cin>>name;

```

```

            cout<<"Age";

```

```

            cin>>age;

```

```

            cout<<"sex="";

```

```

            cin>>sex;

```

```

        }

```

```

void display()

```

```

{

```

```

    cout<<"name =" << name <<endl;

```

```

    cout<<"Age=" <<Age<<endl;

```

```

    cout<<"sex=" <<sex<<endl;

```

```

}

```

```

};

```

```

class student:Public Person

```

```

{

```

```

    private:

```

```

        int rollno;

```

```

        char branch[50];

```

```

    public:

```

```

        getdata()

```

```

        {

```

```

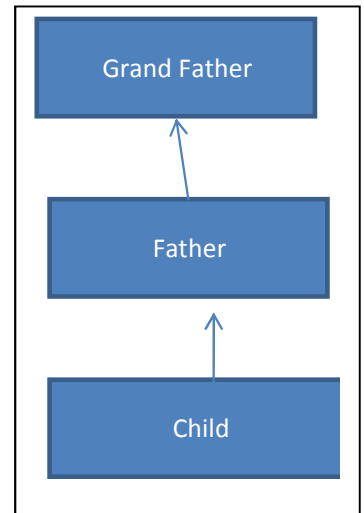
            Person::getdata();

```

```

            cout<<"Rollno="";

```



```

        cin>>rollno;
        cout<<"Branch";
        cin>>branch;
    };

void display()
{
    person:: display();
    cout<<"Rollno ="<<rollno<<endl;
    cout<<"Branch="<<branch<<endl;
}
};
class Exam : public Student
{
    protected:
        int marks1;
        int marks2;
    public:
        void getdata()
        {
            Student:: getdata();
            cout<<"Marks 1=";
            cin>>marks1;
            cout<<"Marks2=";
            cin>>marks2;
        }
}

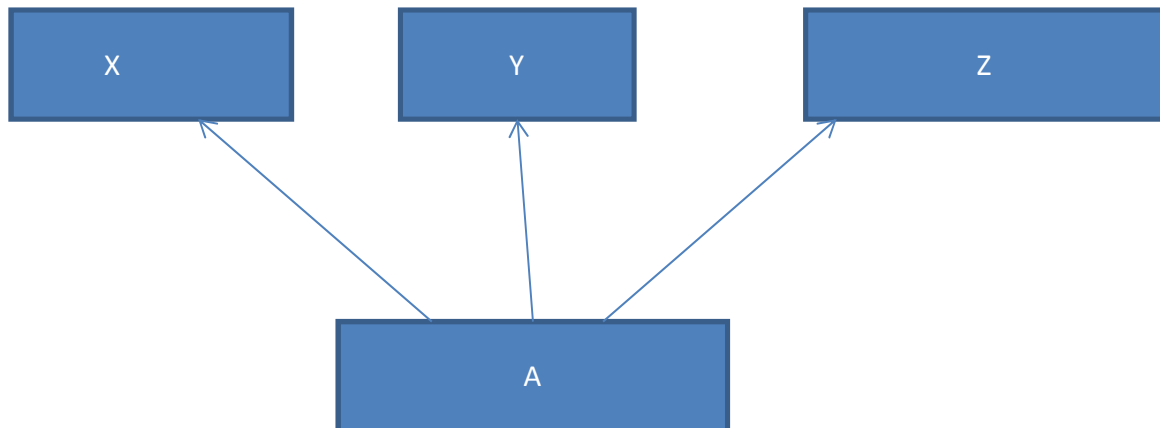
void display()
{
    Student:: display()
    cout<<"Marks1="<<marks1<<endl;
    cout<<"Marks2="<<marks2<<endl;
    cout<<"Total Marks="<<Totalmarks();
}
int Totalmarks()
{
    return marks1+marks2;
}
};
void main()
{
    clrscr();
    Exam e;
    e.getdata();
    e.display();

}

```

Multiple Inheritance: A class can be derived from more than one base class i.e a class can inherit the properties of two or more than two classes. This is called multiple inheritance.

Base Classes



Syntax:-

class derivedclass : Visibility BaseClass1, Visibility Base Class2,

E.g.

```
class X
{
    -----
    -----
};

class y
{
    -----
    -----
};

class Z
{
    -----
    -----
};
```

class A : Public X, Public Y, Public Z

Program: To demonstrate multiple inheritance

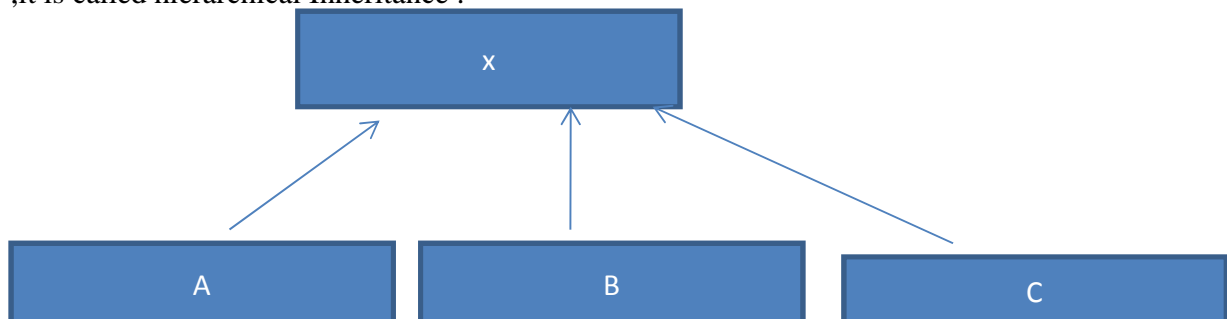
```
#include<iostream.h>
#include<conio.h>
class X
{
    protected:
        int a;
    public:
        void getdata(int);
};
class Y
{
```

```

        protected :
            int b;
        public:
            void getdata1(int);
    };
class Z: public X, public Y
{
    public:
        void display(void);
};
void X::getdata(int m)
{
    a=m;
}
void Y:: getdata1(int n)
{
    b=n;
}
void Z::display(void)
{
    cout<<"a="<<a<<endl;
    cout<<"b="<<b<<endl;
    cout<<"a+b="<<a+b<<endl;
    cout<<"a/b="<<a/b<<endl;
}
void main()
{
    clrscr();
    Z d;
    d.getdata(30);
    d.getdata1(15);
    d.display();
    getch();
}

```

Hierarchical Inheritance: When many derived classes are inherited from single base class ,it is called hierarchical Inheritance .



Syntax:-

```

class A: public X
class B : public X
class C: public X

```

```

Example: #include<iostream.h>
#include< conio.h>
class personal
{
    private:
        int rollno;
        char name[15];
        char sex;
    public:
        void getdata();
        void putdata();
};

class physical : public personal
{
    private:
        float height ;
        float weight ;
    public:
        void getdata();
        void putdata();
};

```

```

class academic : public personal
{
    private:
        char trade[20];
        int semes;
    public:
        void getdata();
        void putdata();
};

void personal :: getdata()
{
    cout<<" Students's name :"; cin>>name;
    cout<<"Rollno:"; cin>> rollno;
    cout<<"Sex:"; cin>>sex;
}

void personal:: putdata()
{
    cout<<"Student's name :"<<name <<endl;
    cout<<"Rollno:"<<rollno<<endl;
    cout<<"Sex:"<<sex<<endl;
}

void physical :: getdata()
{
    personal :: getdata();
    cout <<"Height :"; cin>>height;
}

```

```

        cout<<"Weight:"; cin>> weight;
    }
void physical :: putdata()
{
    personal :: putdata();
    cout<<"Height :"<<height << endl;
    cout<<"Weiight:"<<weight<<endl;    }
void academic :: getdata()
{
    personal :: getdata();
    cout<<" Trade :";
    cin>> trade;
    cout<<"Semester:";
    cin>>semes;    }
void academic :: putdata()
{
    personal :: putdata();
    cout<<" Trade :"<< trade <<endl;
    cout<<"Semester :" << semes <<endl;    }
void main()
{
    class physical obj1;
    class academic obj2;
    clrscr();
    cout<<" ****enter the physical informatioon ****"<<endl;
    obj1.getdata();
    cout<<" ***Enter the academic information ****"<<endl;
    obj2.getdata();
    cout<<endl<<"***** The entered data is *****"<<endl;
    cout<<endl<<"physical fitness "<<endl;
    obj1.putdata();
    cout<<endl<<"Academic Fitness "<<endl;
    obj2.putdata();
    getch();
}

```

Inheritance

SOME IMPORTANT QUESTIONS TO REFRESH THE CONCEPT

- Q1.** Write the reasons behind the introduction of inheritance in OOP.
Ans. The major reasons behind the introduction of inheritance in OOP are:(i) It ensures the closeness with the real world models, (ii) idea of reusability, (iii) transitive nature of inheritance.
- Q2.** What are the different forms of inheritance?
Ans. The different forms of inheritance are (i) Single Inheritance, (ii) Multiple Inheritance, (iii) Hierarchical Inheritance, (iv) Multilevel Inheritance, (v) Hybrid Inheritance.
- Q3.** How does the access of inherited members depend upon their access specifiers and the visibility modes of the base class?
Ans.

Access specifier \ Visibility mode	public inheritance	protected inheritance	private inheritance
public member in base class	public in derived class	protected in derived class	private in derived class
protected member in base class	protected in derived class	protected in derived class	private in derived class
private member in base class	hidden in derived class	hidden in derived class	hidden in derived class

Q4. Write the different ways of accessibility of base class members.

Ans.

Access Specifier	Accessible from own class	Accessible from derived class (Inheritable)	Accessible from objects outside class
public	Yes	Yes	Yes
protected	Yes	Yes	No
private	Yes	No	No

Q5. How is the size of a derived class object calculated?

Ans. The size of a derived class object is equal to the sum of sizes of data members in base class and the derived class.

Q6. In what order are class constructors and class destructors called when a derived class object is created or destroyed?

Ans. When the object of a derived class is created, firstly the constructor of the base class is invoked and then, the constructor of the derived class is invoked.

On the other hand, when the object of a derived class is destroyed, firstly the destructor of the derived class is invoked and then, the destructor of the base class is invoked.

Long Answer Questions (4 Marks)

Q1. Answer the questions (i) to (iv) based on the following:

```
class vehicle
{
    int wheels;
protected:
    int passenger;
public:
    void inputdata();
    void outputdata();
};
```

```
class heavyvehicle : protected vehicle
{
    int diesel_petrol;
protected:
    int load;
public:
    void readdata(int, int);
    void writedata();
};
```

```
class bus : private heavyvehicle
{
    char make[20];
public:
    void fetchdata();
    void displaydata();
};
```

i) Write the member(s) that can be accessed from the object of bus.

ii) Write the data member(s) that can be accessed from the function displaydata().

iii) How many bytes are required by an object of bus and heavyvehicle classes respectively?

iv) Is the member function outputdata() accessible to the objects of the class heavyvehicle?

Ans : (i) fetchdata(), displaydata() (ii) make, load, passenger (iii) for the object of bus – 28 bytes, for the object of heavyvehicle – 8 bytes (iv) No

Q2. Answer the questions (i) to (iv) based on the following:

```
class livingbeing
{
    char specification[20];
    int averageage;
public:
    void read();
    void show();
};
```

```
class ape : private livingbeing
{
    int no_of_organ;
    int no_of_bones;
protected:
    int iq_level;
public:
    void readape();
    void showape();
};
```

```
class human : public ape
{
    char race[20];
    char habitation[30];
public:
    void readhuman();
    void showhuman();
};
```


- (i) Write the members which can be accessed from the member functions of class human.
- (ii) Write the members, which can be accessed by an object of class human.
- (iii) What is the size of an object (in bytes) of class human?
- (iv) Write the class(es) which objects can access read() declared in livingbeing class.

Ans : (i) race, habitation, iq_level, readhuman(), showhuman(), readape(), showape()
(ii) readhuman(), showhuman(), readape(), showape()
(iii) 78 bytes (iv) livingbeing

Q3. Answer the questions (i) to (iv) based on the following:

<pre>class parent { char name[20]; protected: int son; public: void inputdata(); void outputdata(); };</pre>	<pre>class father : protected parent { int daughter; protected: int baby; public: void readdata(); void writedata(); };</pre>	<pre>class mother : public father { int child; public: void fetchdata(); void dispdata(); };</pre>
--	---	--

- (i) In case of the class father, what is the base class of father and what is the derived class of father?
- (ii) Write the data member(s) that can be accessed from function dispdata().
- (iii) Write the member function(s), which can be accessed by an object of mother class.
- (iv) Is the member function outputdata() accessible to the objects of father class?

Ans : (i) base class of father – parent, derived class of father – mother
(ii) child, baby, son
(iii) fetchdata(), dispdata(), readdata(), writedata() (iv) No

Q4. Answer the questions (i) to (iv) based on the following:

<pre>class person { char name[20], address[20]; protected: int x; public: void enter_person(); void disp_person(); };</pre>	<pre>class client : private person { int resource; public: int get_resource(); void free_resource(); };</pre>	<pre>class doctor : public person { char speciality[20]; public: void input(); void disp(); };</pre>
---	---	--

- (i) What type of inheritance is depicted by the above example?
- (ii) Write the member functions, which can be called by the object of class client.
- (iii) What is the size in bytes of the object of class doctor and client respectively?
- (iv) Write the data members, which can be used by the member functions of the class doctor.

Ans : (i) Hierarchical Inheritance (ii) get_resource(), free_resource()
(iii) size of object of class doctor = 62 and client = 44 (iv) speciality, x

Q5. Answer the questions (i) to (iv) based on the following:

<pre>class author { char name[12]; double royalty; protected: void register(); public: author{}; void enter(); void display(); };</pre>	<pre>class person { char address[20]; protected: float salary; public: person{}; void havelt(); void givelt(); };</pre>	<pre>class employee : private author, public person { int ecode; char dept[30]; public: employee{}; void start(); void show(); };</pre>
---	---	---

65

- (i) Write the names of data members, which are accessible from object of class employee.
- (ii) Write the names of all the member functions which are accessible from object of class person.
- (iii) Write the data members which are accessible from member functions of class employee.
- (iv) How many bytes are required by an object belonging to class employee?

Ans : (i) Nil (ii) haveit(), giveit() (iii) ecode, dept, salary (iv) 76 bytes

Q6. Answer the questions (i) to (iv) based on the following:

<pre>class PUBLISHER { char Pub[12]; double Turnover; protected: void Register(); public: PUBLISHER(); void Enter(); void Display(); };</pre>	<pre>class BRANCH { char CITY[20]; protected: float Employees; public: BRANCH(); void Haveit(); void Giveit(); };</pre>	<pre>class AUTHOR: private BRANCH, public PUBLISHER { int Acode; char Aname[20]; float Amount; public: AUTHOR(); void Start(); void Show(); };</pre>
---	---	--

- (i) Write the names of data members, which are accessible from objects belonging to class AUTHOR.
- (ii) Write the names of all the members which are accessible from member functions of class AUTHOR.
- (iii) How many bytes are required by an object belonging to class AUTHOR?
- (iv) Write the sequence of the constructors' invocation when the object of author is created.

Ans: (i) Nil (ii) Acode, Aname, Amount, Employees, Start(), Show(), Haveit(), Giveit(), Register(), Enter(), Display() (iii) 70 bytes (iv) BRANCH(), PUBLISHER(), AUTHOR()

Q7. Answer the questions (i) to (iv) based on the following:

<pre>class CUSTOMER { int Cust_no; char Cust_Name[20]; protected: void Register(); public: CUSTOMER(); void Status(); };</pre>	<pre>class SALESMAN { int Salesman_no; char Salesman_Name[20]; protected: float Salary; public: SALESMAN(); void Enter(); void Show(); };</pre>	<pre>class SHOP : private CUSTOMER , public SALESMAN { char Voucher_No[10]; char Sales_Date[8]; public: SHOP(); void Sales_Entry(); void Sales_Detail(); };</pre>
--	---	---

- (i) Write the names of data members which are accessible from objects belonging to class CUSTOMER.
- (ii) Write the member functions that are accessible from objects belonging to class SALESMAN.
- (iii) Write the names of all the members which are accessible from member functions of class SHOP.
- (iv) How many bytes will be required by an object belonging to class SHOP?

Ans : (i) None of data members are accessible from objects belonging to class CUSTOMER.

(ii) Enter(), Show()

(iii) Data members: Voucher_No, Sales_Date, Salary

Member function: Sales_Entry(), Sales_Details(), Enter(), Show(), Register(), Status()

(iv) 66 bytes

Function Overloading

Meaning :- Function overloading is the method for calling several functions having the same name. These functions are differentiated during the calling process by the number and types of arguments passed to these functions.

Example:

```
int Add (int );
float Add (float);
int Add (int ,int)
float Add (float ,float)
float Add (int ,float)
```

Declaration and Definition of Overloaded functions:-

```
data_type function_name (arguments .....);
```

Function 's Signature :-

1. value return by the function/return type
2. Name of function
3. Argument list(No of Argument + Type of Argument)

Example:-

```
void sample (int a);
void sample (float b)
void sample (char c);

void sample(int a)
{
cout<<" Entered integer is :"<< a;
}
void sample (float b)
{
cout<<" Entered float number is :"<<b;
}
void sample (char c)
{
cout<<" Entered character is :"<<c;
}
```

Example:-

```
#include<iostream.h>
#include<conio.h>
class demo
{
private:
int a1,a2;
float b1,b2;
double d1,d2;
public:
void getdata();
int cube(int);
float cube(float);
double cube(double);
void display();
```

```

        };

void demo:: getdata()
{
    cout<<"Enter an integer :"; cin >> a1;
    cout<<"Enter a float no: "; cin>> b1;
    cout<<"Enter a double no :"; cin>> d1;
}
int demo :: cube (int a1)
{
    return (a1*a1*a1);
}
float demo:: cube (float b1)
{
    return (b1*b1*b1*);
}
double demo :: cube(double d1)
{
    return(d1*d1*d1);
}
void main()
{
    demo obj;
    a2=obj.cube(a1);
    b2=obj.cube(b1);
    d2=obj.cube(d1);
    cout<<a2<<b2<<d2;
    getch();
}

```

DATA FILE HANDLING IN C++

Key Facts:

- Text file: A text file stores information in readable and printable form. Each line of text is terminated with an **EOL** (End of Line) character.
- Binary file: A binary file contains information in the non-readable form i.e. in the same format in which it is held in memory.
- Stream: A stream is a general term used to name flow of data. Different streams are used to represent different kinds of data flow.
- There are three file I/O classes used for file read / write operations.
 - **ifstream** - can be used for read operations.
 - **ofstream** - can be used for write operations.
 - **fstream** - can be used for both read & write operations.
- **fstream.h:**
 - This header includes the definitions for the stream classes ifstream, ofstream andfstream. In C++ **file input output** facilities implemented through fstream.h header file.
 - It contain predefines set of operation for handling file related input and output fstream.h class ties a file to the program for input and output operation.
 - A file can be opened using:
 - By the constructor of the stream. This method is preferred when single file is used with the stream. (only for input / only for output)
 - By the open() function of the stream.
 - **File modes:**
 - **ios::out** - It creates file in output mode and allows writing into the file.
 - **ios::in** - It creates file in input mode and permit reading from the file.
 - **ios::app** - To retain the previous contents of the file and to append to the end of existing file.
 - **ios::ate** - To place the file pointer at the end of the file, but you can write data any where in the file.
 - **ios::trunc** - It truncates the existing file (empties the file).
 - **ios::nocreate** - If file does not exist this file mode ensures that no file is created and open() fails.
 - **ios::noreplace** - If file does not exist, a new file gets created but if the file already exists, the open() fails.
 - **ios::binary** – Opens a file in binary.

eof(): This function determines the end-of-file by returning true for end of file otherwise returning false.

open(): If you want to manage multiple file with same stream use open().

```
Stream_object.open("Filename",(Filemode));
```

e.g., fstream fio;

```
fio.open("book.dat", ios::out | ios::in | ios::binary);
```

close(): This function terminates the connection between the file and stream associated with it.

```
Stream_object.close();
```

read(): The read() function reads a fixed number of bytes from the specified stream and puts them in the buffer.

*Stream_object.read((char *)& Object, sizeof(Object));*

write(): The write() function writes fixed number of bytes from a specific memory location to the specified stream.

*Stream_object.write((char *)& Object, sizeof(Object));*

Note:

Both functions take two arguments.

- The first is the address of variable, and the second is the length of that variable in bytes. The address of variable must be type cast to type char*(pointer to character type)
- The data written to a file using write() can only be read accurately using read().

get pointer: A get pointer indicates the position in the file at which the next input is to occur.

put pointer: It indicates the position in the file at which the next output is to be placed.

seekg(): It places the get pointer to the specified position in a stream.

seekp(): It places the put pointer to the specified position in a stream.

tellg(): This function returns the current position of the get pointer in a stream.

tellp(): This function returns the current position of the put pointer in a stream.

Steps To Process A File

- Determine the type of link required.
- Declare a stream for the desired types of link.
- Attach the desired file to the declared stream.
- Process the file.
- Close the file link with stream.

General program structure used for operating a Text File

1. Write a function in a C++ to count the number of lowercase alphabets present in a text file "BOOK.txt".

```
int countalpha()
{
    ifstream Fin("BOOK.txt");
    char ch;
    int count=0;
    while(!Fin.eof())
    {
        Fin.get(ch);
        if (islower(ch))
            count++;
    }
    Fin.close();
    return count;
}
```

2. Write a function in C++ to count and display the number of lines starting with alphabet 'A' present in a text file "LINES.TXT".

```
void counter()
{
    char Aline[80];
```

```

int Count=0;
ifstream Fin (“LINES.TXT”);
while(!fin.eof())
{
Fin.getline(Aline,80, ‘\n’)
if (Aline[0]== ‘A’)
Count++;
}
cout<<Count<<endl;
Fin.close( );
}

```

3. Write a function COUNT_DO() in C++ to count the presence of a word ‘do’ in a text file “MEMO.TXT”.

```

void COUNT_DO ()
{
fstream f;
Clrscr();
f.open (“MEMO.TXT”, ios :: in);
char are [80];
char ch;
int i = 0, sum = 0,n = 0 ;
While (f)
{
f. get (ch);
are [i] = ch;
i++;
if (strcpy (ch, “do”))
{ i–;
sum = sum + i;
n++;
}
}
cout<<“The total no. of the do is:”<<n;
}

```

4. Write a function in C++ to count the number of vowels present in a text file “STORY.TXT”.

```

Ans: void vowelcount( )
{
ifstream file(“STORY.TXT”);
int n=0;
while(file.get(ch))
{
if(ch==’a’|| ch==’A’|| ch==’e’|| ch==’E’|| ch==’i’|| ch==’I’|| ch==’o’|| ch==’O’||
ch==’u’||ch==’U’)
n++;
}
cout<<“\n Total no. of vowels are”<<n;
file.close();
}

```

5. **Write a function in C++ to count the number of words in a text file NOTES.TXT.**

```
Ans: void wordcount( )
{
ifstream f1("NOTES.TXT");
char ch;
int tot=0;
f1.seekg(0);
while(f1.get(ch))
{
    if( ch==' ' || ch=='.' || ch=='?')
        tot++;
}
cout<<"\n total no. of words are "<<tot;
f1.close();
}
```

6. **Write a function in C++ to count the number of lines present in a text file "STORY.TXT".**

```
Ans : void CountLine()
{
    ifstream FIL("STORY.TXT");
    int LINES=0;
    char STR[80];
    while (FIL.getline(STR,80))
        LINES++;
    cout<<"No. of Lines:"<<LINES<<endl;
    FIL.close();
}
```

7 **Write a function in C++ to count the word "this" /"THIS" present in Text file "Diary.TXT"**

```
Ans : void wordcount( )
Fstream file;
file.open("Diary.TXT",ios::in);
char word[20];
int count=0;
while (!file.eof( ))
{
file>>word;
if(!strcmp(word,"this")||strcmp(word,"THIS")||!strcmp(word,"THIS"))
count++;
}
cout<<" Number of this/This/THIS="<<count<<endl;
file. Close( ); }
```

8) **Write a function in C++ to read the content of a text file "PLACES.TXT" and display all those lines on screen, which are either starting with 'P' or starting with 'S'.**

```
Ans void DispPorS ( )
{
```



```

ifstream File ("PLACES.TXT");
char STR[80];
while(File.getline(STR,80))
{
if(STR[0]=='P' || STR[0]=='S')
cout<<STR<<endl;
}
File.close(); //Ignore

```

Exercise:

1. Create a function in a C++ to count the number of uppercase alphabets present in a text file "BOOK.txt"
2. Write a function in a C++ to count the number of alphabets present in a text file "BOOK.txt"
3. Write a function in a C++ to count the number of digits present in a text file "BOOK.txt"
4. Write a function in a C++ to count the number of white spaces present in a text file "BOOK.txt"
5. Write a function in a C++ to count the number of vowels present in a text file "BOOK.txt"
6. Write a function in a C++ to count the average word size in a text file "BOOK.txt"
7. Write a function in C++ to print the count of the word "the" as an independent word in a text file STORY.TXT.

For example, if the content of the file STORY.TXT is

There was a monkey in the zoo.

The monkey was very naughty.

Then the output of the program should be 2.

8. Assume a text file "Test.txt" is already created. Using this file, write a function to create three files "LOWER.TXT" which contains all the lowercase vowels and "UPPER.TXT" which contains all the uppercase vowels and "DIGIT.TXT" which contains all digits.
9. Create a function FileLowerShow() in c++ which take file name(text files)as a argument and display its all data into lower case
10. Write a function in C++ to count the number of lines present in a text file "Story.txt".

HOTS QUESTION BASED ON FILE HANDLING

1. Write a function in a C++ to count the number of consonants present in a text file "Try.txt"
2. Write a function in a C++ to count the number of uppercase vowels present in a text file "Novel.txt"
3. Write a function in a C++ to display the sum of digits present in a text file "Fees.txt".
4. Write a function in a C++ to display the product of digits present in a text file "Number.txt".
5. Write a function in a C++ to find the largest digit present in a text file "Marks.txt"

General program structure used for operating a Binary File

- Q1. Write a function in C++ to add more new objects at the bottom of a binary file "STUDENT.DAT", assuming the binary file is containing the objects of the following class.

```
class STUD
{
    int Rno;
    char Name[20];
public:
    void Enter(){cin>>Rno;gets(Name);}
    void Display(){cout<<Rno<<Name<<endl;}
};
```

```
Ans: void Addnew()
{
    ofstream FIL;
    FIL.open("STUDENT.DAT", ios::out | ios::binary | ios::app);
    STUD S;
    char CH = 'y';
    while(CH == 'Y' || CH == 'y')
    {
        S.Enter();
        FIL.write((char*)&S,sizeof(S));
        cout<<"More(Y/N)?">>cin>>CH;
    }
    FIL.close();
}
```

- Q2. Write a function in C++ to read & display the details of all the members whose membership type is 'L' or 'M' from a binary file "CLUB.DAT". Assume the binary file contains object of class CLUB.

```
class CLUB
{
    int mno;
    char mname[20];
    char type;
public:
    void register ();
    void dis();
    char whattype()
    { return type;}
};
Ans : void show()
{
    ifstream file;
    CLUB c1;
    char s;
```

```

file.open("CLUB.DAT", ios::in | ios::binary);
while( file.read((char*)&c1, sizeof (c1)))
{
    s = c1.whattype( );
    if((s=='L') || (s=='M'))
        c1.dis( );
} File.close(); }

```

Q3. Given a binary file PHONE.DAT containing records of the following structure type.class Phonlist

```

{
    char add;
    char Name[20];
    char area[5];
public:
    void register();
    void show();
    int check(char ac[ ])
    {
        return strcmp(area,ac)
    }
};

```

Write a function TRANSFER() in C++, that would copy all records which are having area as "DEL" from PHONE.DAT to PHONBACK.DAT.

Ans:

```

void transfer( )
{
    ifstream Fin;
    ofstream Fout;
    Phonlist ph;
    Fin.open("PHONE.DAT", ios::in | ios::binary);
    Fout.open("PHONBACK.DAT", ios::out | ios:: binary);
    while(Fin.read((char*)&ph, sizeof(ph)))
    {
        if(ph.check("DEL") == 0)
            Fout.write((char*)&ph, sizeof(ph));
    }
    Fin.close();
    Fout.close();
}

```

Q4. Write a function in C++ to modify the name of a student whose roll number is entered during the execution of the program. Search the accepted roll number from a binary file "STUDENT.DAT" and modify the name, assuming the binary file is containing the objects of the following class.

```

class STUD
{

```

```

        int Rno;
        char Name[20];
    public:
        int srno(){ return Rno;}
        void Enter(){ gets(Name);}
        void Display(){ cout<<Rno<<Name<<endl;}
};

```

```

Ans : void Modifyname()
{
    fstream File;
    File.open("STUDENT.DAT", ios::binary | ios::in | ios::out);
    STUD s;
    int mrno;
    cout<<"\nEnter the roll number to modify : ";
    cin>>mrno;
    while (File.read((char*)&s, sizeof(s)))
    {
        if(s.srno() == mrno)
        {
            cout<<"\n Modify the name ";
            s.Enter();
            File.seekp( -1 * sizeof(s), ios::cur);
            File.write((char *)&s, sizeof(s));
        }
    }
    File.close ();
}

```

5. Write a function in C++ to read and display the detail of all the users whose status is 'A' (i.e. Active) from a binary file "USER.DAT". Assuming the binary file "USER.DAT" is containing objects of class USER, which is defined as follows :

```

class USER
{
    int Uid; //User Id
    char Uname [20]; //User Name
    char Status; // User Type : A Active I Inactive
    public:
    void Register(); //Function to enter the content
    void show(); //Function to display all data members
    char Getstatus () {return Status;}.
};

```

Answer

```

void fun()
{
    fstream FILE;
    File. open ("USER.DAT") ios ::binary|ios :: in);
    USER U;
    while (FILE. read ((char*)&U,sizeof (U)))
    if (U. Getstatus ()== 'A')

```

```

U. show()
FILE. close ();
}

```

6. Given a binary file STUDENT.DAT, containing records of the following class

Student type

```

class Student
{
char S_Adjno[10]; //Admission number of student
char S_Name[30]; //Name of student
int Percentage; //Marks Percentage of student
public:
void EnterData()
{
gets(S_Adjno);gets(S_Name);cin>>Percentage;
}
void DisplayData()
{
cout<<setw(12)<<S_Adjno;
cout<<setw(32)<<S_Name;
cout<<setw(3)<<Percentage<<endl;
}
int ReturnPercentage(){return Percentage;}
};

```

Write a function in C++, that would read contents of file STUDENT.DAT and display the details of those Students whose Percentage is above

```

Answer :- void Distinction()
{
Student S;
fstream Fin;
Fin.open("STUDENT.DAT", ios::binary|ios::in);
while(Fin.read((char*)&S, sizeof(Student))
if (S.ReturnPercentage(>75)
S.DisplayData();
Fin.close();
}

```

7. Given a binary file APPLY.DAT, containing records of the following class

Applicant

type

```

class Applicant
{
char A_Rno[10]; //Roll number of applicant
char A_Name[30]; //Name of applicant
int A_Score; //Score of applicant
public:
void Enrol()
{
gets(A_Rno); gets(A_Name) ; cin>>A_Score;
}
}

```

```

}
void Status()
{
cout<<setw(12)<<A_Admno;
cout<<setw(32)<<A_Name;
cout<<setw(3)<<A_Score<<endl;
}
int ReturnScore(){return A_Score;}
};

```

Write a function in C++, that would read contents of file APPLY.DAT and display the Details of those Students whose A_Score is below 70.

```

Answer :-void READAPPLY() //Ignore
{
fstream FILE;
FILE.open("APPLY.DAT",ios::binary|ios::in);
Applicant A;
while (FILE.read((char*)&A,sizeof(A)))
if (A.ReturnScore(<70)
A.Status();
FILE.close(); //Ignore
}

```

Questions based on file pointer

1. Observe the program segment given below carefully and fill the blanks marked as Line 1 and Line 2 using fstream functions for performing the required task. 1

```

#include <fstream.h>
class Library
{
long Ano; //Ano – Accession Number of the Book
char Title[20]; //Title – Title of the Book
int Qty; //Qty – Number of Books in Library
public:
void Enter(int); //Function to enter the content
void Display(); //Function of display the content
void Buy(int Tqty)
{
Qty+=Tqty;
} //Function to increment in Qty
long GetAno() {return Ano;}
};
void BuyBook (long BANo, int BQty)
//BANo @ Ano of the book purchased
//BQty @ Number of books purchased
{
fstream File;
File. open ("STOCK.DAT", ios: : binary|ios: : in|ios: : out);
int Position=-1;
Library L;

```

```

while (Position == -1 && File.read ((char*) &L, sizeof (L)))
if (L.GetAno() == BANo)
{
L.Buy (BQty); //To update the number of Books
Positions=File.tellg()-sizeof (L);
//Line 1: To place the file pointer to the required position.
_____
//Line 2: To write the object L on to the binary file
_____
}
if (Position===-1)
cout<<"No updation done as required Ano not found...";
File.Close();
}
File.seekp (position, ios :: beg); // Line-1
File.write ((char *) & L, sizeof (L)); // Line-2

```

2. Observe the program segment given below carefully and fill the blanks marked as Statement 1 and Statement 2 using seekg() and tellg() functions for performing the required task.

```

#include <fstream.h>
class Employee
{
int Eno;char Ename[20];
public:
//Function to count the total number of records
int Countrec();
};
int Item::Countrec()
{
fstream File;
File.open("EMP.DAT",ios::binary|ios::in);
_____
//Statement 1

int Bytes = _____ //Statement 2

int Count = Bytes / sizeof(Item);
File.close();
return Count;
}

```

Answer:

<pre> File.seekg(0,ios::end);//Statement 1 File.tellg(); //Statement 2 </pre>

3. Observe the program segment given below carefully and fill the blanks marked as Line 1 and Line 2 using fstream functions for performing the required task.

```

#include <fstream.h>

```

```

class Stock
{
long Ino; //Item Number
char Item [20]; //Item Name
int Qty; //Quantity
public:
void Get(int); //Function to enter the content
void show(); //Function to display the content
void Purchase (int Tqty)
{
Qty+=Tqty;
} //Function to increment in Qty
long KnowIno () {return Ino;}
};
void Purchaseitem (long PINo, int PQty)
//PINo -> Ino of the item purchased
//PQty -> Number of item purchased
{
fstream File;
Solved Paper, 2009 | 5
File.open ("ITEMS.DAT", ios :: binary|ios :: in|ios :: out);
int Pos=-1;
Stock S;
while (Pos== -1 && File.read ((char*) &L, sizeof (S)))
if (S. KnowIno()==PINO)
{
S. Purchase (PQty); //To update the number of Items
Pos=File.tellg () -sizeof (S);
//Line 1: To place the file pointer to the required position.
_____
;
//Line 2: To write the object S on to the binary file
_____
;
}
if (Pos== -1
cout<<"No updation done as required Ino not found.";
File.close();
}

```

Answer :-

File. seekg (pos * sizeof (item)); //statement

File write ((char*) &L, size of<<); //statement2

Practice Work

1.The program segment carefully and answer the question that follows:

```

class item
{
int item_no;
char item_name[20];
public:
void enterDetail( );

```



```

void showDetail( );
int getItem_no( ){ return item_no;}
};
void modify(item x, int y )
{
fstream File;
File.open( "item.dat", ios::binary | ios::in | ios::out) ;
item i;
int recordsRead = 0, found = 0;
while(!found && File.read((char*) &i , sizeof (i)))
{
recordsRead++;
if(i . getItem_no( ) == y )
{
_____//Missing statement
File.write((char*) &x , sizeof (x));
found = 1;
}
}
if(! found)
cout<<"Record for modification does not exist" ;
File.close() ;
}

```

If the function modify() is supposed to modify a record in the file “ item.dat “, which item_no is y, with the values of item x passed as argument, write the appropriate statement for the missing statement using seekp() or seekg(), whichever is needed, in the above code that would write the modified record at its proper place.

2. Observe the program segment carefully and answer the question that follows:

class item

```

{int item_no;
char item_name[20];
public:
void enterDetails( );
void showDetail( );
int getItem_no( ){ return item_no;}
};
void modify(item x )
{fstream File;
File.open( "item.dat", _____ ) ; //parameter missing
item i;
while(File .read((char*) & i , sizeof (i)))
{if(x . getItem_no( ) == i . getItem_no( ))
{File.seekp(File.tellg( ) – sizeof(i));
File.write((char*) &x , sizeof (x));
}
else
File.write((char*) &i , sizeof (i));
}
}

```

```
File.close() ;
}
```

If the function modify () modifies a record in the file “ item.dat “ with the values of item x passed as argument, write the appropriate parameter for the missing parameter in the above code, so as to modify record at its proper place.

03. Observe the program segment carefully and answer the question that follows:

class item

```
{
int item_no;
char item_name[20];
public:
void enterDetail( );
void showDetail( );
int getItem_no(){ return item_no;}
};
void modify(item x )
{
fstream File;
File.open( “item.dat”, ios::binary|ios::in|ios::out ) ;
item i;
while(File .read((char*) & i , sizeof(i))//Statement 1
{
if(x . getItem_no( ) == i . getItem_no( ))
{
File.seekp(File.tellg( ) – sizeof(i));
File.write((char*) &x , sizeof (x));
}
}
File.close() ;
}
```

If the function modify() modifies a record in the file “ item.dat” with the values of item x passed as argument, rewrite statement 1 in the above code using eof(), so as to modify record at its proper place.

4.Observe the program segment given below carefully, and answer the question that follows :

```
class Member
{
int Member_no;
char Member_name[20];
public :
//function to enter Member details
void enterdetails( ) ;
// function to display Member details
void showdetails();
//function to return Member_no
int RMember_no( ) {return Member_no; }
};
void Update(Member NEW)
```

```

{
fstream File;
File.open("MEMBER.DAT",ios::binary|ios::in|ios::out);
Member OM;
int Recordsread = 0, Found = 0;
while (!Found && File.read((char*)&OM, sizeof(OM)))
{
Recordsread ++;
if (NEW.RMember_no() == OM.RMember_no())
{
_____//Missing Statement
File.write((char*)&NEW, sizeof(NEW));
Found = 1;
}
else
File.write((char*)&OM, sizeof(OM));
}
if (!Found)
cout<<"Record for modification does not exist";
File.close();
}

```

If the function Update () is supposed to modify a record in file MEMBER.DAT with the values of Member NEW passed to its argument, write the appropriate statement for **Missing Statement** using seekp () or seekg (), whichever needed, in the above code that would write the modified record at its proper place.

POINTERS:

- Pointer is a variable that holds a memory address of another variable of same type.
- It supports dynamic allocation routines.
- It can improve the efficiency of certain routines.

C++Memory Map :

- Program Code : It holds the compiled code of the program.
- Global Variables : They remain in the memory as long as program continues.
- Stack : It is used for holding return addresses at function calls, arguments passed to the functions, local variables for functions. It also stores the current state of the CPU.
- Heap : It is a region of free memory from which chunks of memory are allocated via DMA functions.

Static Memory Allocation : The amount of memory to be allocated is known in advance and it allocated during compilation, it is referred to as Static Memory Allocation.

e.g. int a; // This will allocate 2 bytes for a during compilation.

Dynamic Memory Allocation : The amount of memory to be allocated is not known beforehand rather it is required to allocated as and when required during runtime, it is referred to as dynamic memory allocation.

C++ offers two operator for DMA – **new and delete**.

e.g., int x =new int; float y= new float; // dynamic allocation
 delete x; delete y; //dynamic deallocation

Free Store : It is a pool of unallocated heap memory given to a program that is used by the program for dynamic memory allocation during execution.

Declaration and Initialization of Pointers :

Syntax :

```
Datatype *variable_name;
e.g., int *p;          float *p1;          char *c;
```

Two special unary operator * and & are used with pointers. The & is a unary operator that returns the memory address of its operand.

e.g., int a = 10; int *p; p = &a;

Pointer arithmetic:

Two arithmetic operations, addition and subtraction, may be performed on pointers. When you add 1 to a pointer, you are actually adding the size of whatever the pointer is pointing at. That is, each time a pointer is incremented by 1, it points to the memory location of the next element of its base type.

e.g. int *p; p++;

If current address of p is 1000, then p++ statement will increase p to 1002, not 1001.

Adding 1 to a pointer actually adds the size of pointer's base type.

Base address : A pointer holds the address of the very first byte of the memory location where it is pointing to. The address of the first byte is known as **BASE ADDRESS**.

Dynamic Allocation Operators :

C++ dynamic allocation allocate memory from the free store/heap/pool, the pool of unallocated heap memory provided to the program. C++ defines two unary operators **new** and **delete** that perform the task of allocating and freeing memory during runtime.

Creating Dynamic Array :

Syntax : pointer-variable = new data-type [size];

e.g. int * array = new int[10];

array[0] will refer to the first element of array, array[1] will refer to the second element.

No initializes can be specified for arrays.

All array sizes must be supplied when new is used for array creation.

Two dimensional array :

```
int *arr, r, c;
```

```
r = 5; c = 5;
```

```
arr = new int [r * c];
```

Now to read the element of array, you can use the following loops :

```
For (int i = 0; i < r; i++)
```

```
{
```

```
cout << "\n Enter element in row " << i + 1 << " : ";
```

```
For (int j=0; j < c; j++)
```

```
cin >> arr [ i * c + j];
```

```
}
```

Memory released with delete as below:

Syntax for simple variable : delete pointer-variable;

eg. delete p;

For array :

```
delete [size] pointer variable;
```

e.g. delete [] arr;

Pointers and Arrays :

C++ treats the name of an array as constant pointer which contains base address i.e address of first location of array. Therefore Pointer variables are efficiently used with arrays for declaration as well as accessing elements of arrays, because array is continuous block of same memory locations and therefore pointer arithmetic help to traverse in the array easily.

```
void main()
{
int *m;
int marks[10] = { 50,60,70,80,90,80,80,85,75,95 };
m = marks; // address of first location of array or we can write it as m=&marks[0]
for(int i=0;i<10;i++)
cout<< *m++;
// or
m = marks; // address of first location of array or we can write it as m=&marks[0]
for(int i=0;i<10;i++)
cout<< *(m+i);
}
```

Array of Pointers :

To declare an array holding 10 int pointers –

```
int * ip[10];
```

That would be allocated for 10 pointers that can point to integers.

Now each of the pointers, the elements of pointer array, may be initialized. To assign the address of an integer variable phy to the forth element of the pointer array, we have to write `ip[3] = &phy`; Now with `*ip[3]`, we can find the value of phy. `int *ip[5];`

Index	0	1	2	3	4
address	1000	1002	1004	1006	1008
int a = 12, b = 23, c = 34, d = 45, e = 56;					
Variable	a	b	c	d	e
Value	12	23	34	45	56
address	1050	1065	2001	2450	2725

```
ip[0] = &a; ip[1] = &b; ip[2] = &c; ip[3] = &d; ip[4] = &e;
```

Index	ip[0]	ip[1]	ip[2]	ip[3]	ip[4]
Array ip value	1050	1065	2001	2450	2725
address	1000	1002	1004	1006	1008

ip is now a pointer pointing to its first element of ip. Thus ip is equal to address of ip[0], i.e. 1000

```
*ip (the value of ip[0]) = 1050
```

```
*( * ip) = the value of *ip = 12
```

```
** (ip+3) = ** (1006) = * (2450) = 45
```

Pointers and Strings :

Pointer is very useful to handle the character array also. E.g :

```
void main()
{ char str[] = "computer";
char *cp;
cp=str;
cout<<str ; //display string
cout<<cp; // display string
```

```

for (cp =str; *cp != '\0'; cp++) // display character by character by character
cout << "--"<<*cp;
// arithmetic
str++; // not allowed because str is an array and array name is constant pointer
cp++; // allowed because pointer is a variable
cout<<cp;}

```

Output :

```

Computer
Computer
--c--o--m--p--u--t--e--r
omputer

```

An array of char pointers is very useful for storing strings in memory. Char

```
*subject[] = { "Chemistry", "Phycics", "Maths", "CS", "English" };
```

In the above given declaration subject[] is an array of char pointers whose element pointers contain base addresses of respective names. That is, the element pointer subject[0] stores the base address of string "Chemistry", the element pointer subject[1] stores the above address of string "Physics" and so forth.

An array of pointers makes more efficient use of available memory by consuming lesser number of bytes to store the string.

An array of pointers makes the manipulation of the strings much easier. One can easily exchange the positions of strings in the array using pointers without actually touching their memory locations.

Pointers and CONST :

A constant pointer means that the pointer in consideration will always point to the same address. Its address can not be modified.

A pointer to a constant refers to a pointer which is pointing to a symbolic constant. Look the following example :

```

int m = 20;           // integer m declaration
int *p = &m;         // pointer p to an integer m
++ (*p);             // ok : increments int pointer p
int * const c = &n;  // a const pointer c to an intger n
++ (* c);            // ok : increments int pointer c i.e. its contents
++ c;                // wrong : pointer c is const – address can't be modified
const int cn = 10;   // a const integer cn
const int *pc = &cn; // a pointer to a const int
++ (* pc);           // wrong : int * pc is const – contents can't be modified
++ pc;               // ok : increments pointer pc
const int * const cc = *k; // a const pointer to a const integer
++ (* cc);           // wrong : int *cc is const
++ cc;               // wrong : pointer cc is const

```

Pointers and Functions :

A function may be invoked in one of two ways :

1. call by value
2. call by reference

The second method call by reference can be used in two ways :

1. by passing the references
2. by passing the pointers

Reference is an alias name for a variable. For ex : int m =23;

```
int &n = m;
```

```
int *p;
p = &m;
Then the value of m i.e. 23 is printed in the following ways :
cout << m;           // using variable name
cout << n;           // using reference name
cout << *p;         // using the pointer
```

Invoking Function by Passing the References :

When parameters are passed to the functions by reference, then the formal parameters become references (or aliases) to the actual parameters to the calling function. That means the called function does not create its own copy of original values, rather, it refers to the original values by different names i.e. their references.

For example the program of swapping two variables with reference method :

```
#include<iostream.h>
void main()
{
void swap(int &, int &);
int a = 5, b = 6;
cout << "\n Value of a : " << a << " and b : " << b;
swap(a, b);
cout << "\n After swapping value of a : " << a << "and b : " << b;
}
void swap(int &m, int &n)
{
int temp; temp = m;
m = n;
n = temp;
}
```

output :

Value of a : 5 and b : 6

After swapping value of a : 6 and b : 5

Invoking Function by Passing the Pointers:

When the pointers are passed to the function, the addresses of actual arguments in the calling function are copied in to formal arguments of the called function.

That means using the formal arguments (the addresses of original values) in the called function, we can make changing the actual arguments of the calling function.

For example the program of swapping two variables with Pointers :

```
#include<iostream.h>
void main()
{
void swap(int *m, int *n);
int a = 5, b = 6;
cout << "\n Value of a : " << a << " and b : " << b;
swap(&a, &b);
cout << "\n After swapping value of a : " << a << "and b : " << b;
}
void swap(int *m, int *n)
{
```

```
int temp;
temp = *m;
*m = *n;
*n = temp;
}
```

Input :

Value of a : 5 and b : 6

After swapping value of a : 6 and b : 5

Function returning Pointers :

The way a function can returns an int, an float, it also returns a pointer. The general form of prototype of a function returning a pointer would be

Type * function-name (argument list);

```
#include <iostream.h>
int
*min(int &, int &); void main()
{
int a, b, *c;
cout << "\nEnter a :"; cin >> a;
cout << "\nEnter b :"; cin >> b;
c = min(a, b);
cout << "\n The minimum no is : " << *c;
}
int *min(int &x, int &y)
{
if (x < y)
return (&x);
else
return (&y)
}
```

Dynamic structures :

The new operator can be used to create dynamic structures also i.e. the structures for which the memory is dynamically allocated.

```
struct-pointer = new struct-type;
```

```
student *stu;
```

```
stu = new Student;
```

A dynamic structure can be released using the deallocation operator delete as shown below :

```
delete stu;
```

Objects as Function arguments :

Objects are passed to functions in the same way as any other type of variable is passed. When it is said that objects are passed through the call-by-value, it means that the called function creates a copy of the passed object.

A called function receiving an object as a parameter creates the copy of the object without invoking the constructor. However, when the function terminates, it destroys this copy of the object by invoking its destructor function.

If you want the called function to work with the original object so that there is no need to create and destroy the copy of it, you may pass the reference of the object. Then the called function refers to the original object using its reference or alias. Also the object pointers are declared by placing in front of a object pointer's name.

Classname * object-pointer;

e.g. Student *stu;

The member of a class is accessed by the arrow operator (->) in object pointer method.

e.g :

```
#include<iostream.h>
```

```
class Point
```

```
{
```

```
int x, y;
```

```
public :
```

```
Point()
```

```
{ x = y = 0;}
```

```
void getPoint(int x1, int y1)
```

```
{ x = x1; y = y1; }
```

```
void putPoint()
```

```
{ cout << "\n
```

```
Point :
```

```
(" << x
```

```
<< " , "
```

```
<< y
```

```
<< " );
```

```
}};
```

```
void main()
```

```
{
```

```
Point p1, *p2;
```

```
cout << "\n Set point at 3, 5 with object";
```

```
p1.getPoint(3,5);
```

```
cout << "\n The point is :";
```

```
p1.putPoint();
```

```
p2 = &p1;
```

```
cout << "\n Print point using object pointer :";
```

```
p2->putPoint();
```

```
cout << "\n Set point at 6,7 with object pointer";
```

```
p2->getPoint(6,7);
```

```
cout<< "\n The point is :";
```

```
p2->putPoint();
```

```
cout << "\n Print point using object :";
```

```
p1.getPoint();}
```

If you make an object pointer point to the first object in an array of objects, incrementing the pointer

would make it point to the next object in sequence.

```
student stud[5], *sp;
```

```
---
```

```
sp = stud;           // sp points to the first element (stud[0])of stud
```

```
sp++;               // sp points to the second element (stud[1]) of stud
```

```
sp + = 2;           // sp points to the fourth element (stud[3]) of stud
```

```
sp--;               // sp points to the third element (stud[2]) of stud
```

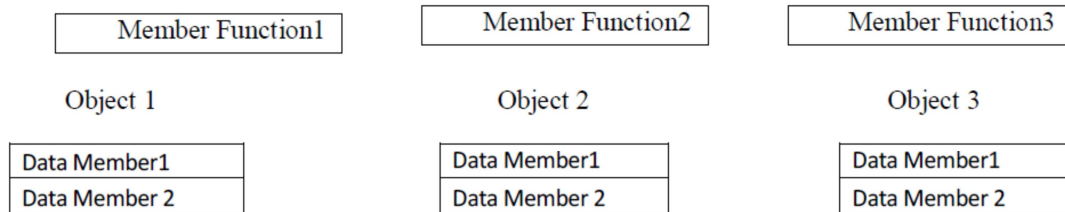
You can even make a pointer point to a data member of an object. Two points should be considered :

1. A Pointer can point to only public members of a class.

2. The data type of the pointer must be the same as that of the data member it points to.

this Pointer :

In class, the member functions are created and placed in the memory space only once. That is only one copy of functions is used by all objects of the class. Therefore if only one instance of a member function exists, how does it come to know which object's data member is to be manipulated?



For the above figure, if Member Function2 is capable of changing the value of Data Member3 and we want to change the value of Data Member3 of Object3. How would the Member Function2 come to know which Object's Data Member3 is to be changed?

To overcome this problem this pointer is used.

When a member function is called, it is automatically passed an implicit argument that is a pointer to the object that invoked the function. This pointer is called This.

That is if object3 is invoking member function2, then an implicit argument is passed to member function2 that points to object3 i.e. this pointer now points to object3.

The friend functions are not members of a class and, therefore, are not passed a this pointer. The static member functions do not have a this pointer.

Unit-II

Data Structure

In Computer Science, a **data structure** is a particular way of storing and organizing data in a computer so that it can be used efficiently. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. For example, Stacks are used in function call during execution of a program, while B-trees are particularly well-suited for implementation of databases. The data structure can be classified into following two types:

Simple Data Structure: These data structures are normally built from primitive data types like integers, floats, characters. For example arrays and structure.

Compound Data Structure: simple data structures can be combined in various ways to form more complex structure called compound structures. Linked Lists, Stack, Queues and Trees are examples of compound data structure.

Array

Array is a collection of elements of similar data type stored in continuous memory location under a single name.

Declaration of array

Syntax:-

Datatype array name[total no of elements];

Eg. int arr[10];

The statement written above will create an array namely arr having 10 integer elements

To differentiate array elements index value is used. Index value is an integer value that starts from 0 and moves upto N-1, where N is total number of elements in the array. For example if we have an array of 10 elements then the index value will be from 0 to 9

Initialization of the array

Datatype array name[no of elements]={ value1,value2,.....valueN};

Eg int arr[5]={ 10,20,30,40,50};

We can skip the total no elements and can initialize the array

Eg int arr[]={ 10,20,30};

Or

int arr[]={ 10,20,30,40,50,60};

Program to enter and display elements in an array

```
void main()
{
  Int arr[10],I;
  cout<<"enter the elements in array"<<endl;
  for(i=0;i<10;i++)
  {
    cin>>arr[i];
  }
  cout<<"The Array elements are"<<endl;
  for(i=0;i<10;i++)
  {
```

```
cout<<arr[i]<<endl;
}
```

Searching methods in array

Linear Search: In this method each element of the array is compared with the number to be searched in linear order (from first to last). And where the number is matched the position is displayed.

```
#include<iostream.h>
#include<conio.h>
void main()
{
int lsearch(int[],int,int);
int a[50],item,n,index;
clrscr();
cout<<"\n Enter size of array";
cin>>n;
cout<<"\n Enter array elements";
for(int i=0;i<n;i++)
cin>>a[i];
cout<<"Enter the item to be searched";
cin>>item;
index=lsearch(a,n,item);
if(index== -1)
cout<<"\n Element not found";
else
cout<<"\n Element found at position "<<index+1;
getch();
}
```

```
int lsearch(int a[],int size,int item)
{
Int found=0;
for(int i=0;i<size;i++)
{
if(a[i]==item)
{
return i;
found=1;
break;
}
}
if(found==0)
return -1;
}
```

Binary Search Method

Binary search algorithm is applicable for already sorted array only. In this algorithm, to search for the given item from the sorted array (in ascending order), the item is compared with the middle element of the array. If the middle element is equal to the item then index of the middle element is returned, otherwise, if item is less than the middle item then the item is present in first

half segment of the array (i.e. between 0 to middle-1), so the next iteration will continue for first half only, if the item is larger than the middle element then the item is present in second half of the array (i.e. between middle+1 to size-1), so the next iteration will continue for second half segment of the array only. The same process continues until either the item is found (search successful) or the segment is reduced to the single element and still the item is not found (search unsuccessful).

```
#include<iostream.h>
#include<conio.h>

void main()
{
int bsearch(int[],int,int);
int a[50], item, n, index;
clrscr();
cout<<"\n Enter total elements";
cin>>n;
cout<<"\n Enter array elements in sorted form:";
for(int i=0;i<n;i++)
cin>>a[i];
cout<<"Enter the item to be searched";
cin>>item;
index=bsearch(a, n, item);
if(index== -1)
cout<<"\n Element not found";
else
cout<<"\n Element found at position "<<index+1;
getch();
}

int bsearch(int a[], int size, int item)
{
int beg, med, last;
beg=0,found=0;
last=size-1;
int mid=(last+beg)/2;
while(beg<=last)
{
mid=(beg+last)/2;
if(item== a[mid])
{
return mid;
found=1;
break;
}
else if(item>a[mid])
beg=mid+1;
else
last=mid-1;
}
```

```

}
If(found==0)
    return -1;
}

```

Sorting operation in the array

Sorting means to arrange the array elements in Ascending order or in Descending order. There are various methods to do this but for the ease sake Bubble sort method is displayed here.

```

#include<iostream.h>
#include<conio.h>
void bubblesort (int[],int);
void main()
{
    int a[50],n;
    clrscr();
    cout<<"\nHow many elements do you want to create array with? ";
    cin>>n;
    cout<<"\nEnter array elements\n";
    for(int i=0;i<n;i++)
        cin>>a[i];
    bubblesort(a,n);
    cout<<"\n\nThe sorted array is as shown below\n";
    for(i=0;i<n;i++)
        cout<<a[i]<<"\n";
    getch();
}
void bubblesort(int a[],int n) //Function to perform bubble sort
{
    int temp;
    for(int i=0;i<n-1;i++)
    {
        for(int j=0;j<n-i-1;j++)
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
    }
}

```

Some questions based on array

Q1. Write a function in C++ which accepts an integer array and its size as arguments/parameters and reverses the array

example : if the array is 1,2,3,4,5 then rearrange the array as 5,4,3,2,1

Ans : void reverse(int arr[], int n)

```

{
    int temp;
    for(int i=0,j=n-1; i<=j; i++,j--)

```

```

    {
    temp= arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
}

```

Q2. Write a function in C++ which accepts an integer array and its size as arguments/parameters and exchange the array in the given manner

example : if the array is 1,2,3,4,5,6,7,8,9,10 then rearrange the array as 2,1,4,3,6,5,8,7,10,9

Ans : void change(int arr[], int n)

```

{
    int temp;
for(int i=0; i<n; i=i+2)
    {
    temp= arr[i];
    arr[i] = arr[i+1];
    arr[i+1] = temp;
}
}

```

Q3 Write a function in C++ to merge the contents of two sorted arrays A & B into third array C. Assuming array A is sorted in ascending order, B is sorted in descending order, the resultant array is required to be in ascending order.

Ans: void MERGE(int A[], int B[], int C[], int M, int N, int &K)

```

{
int I,J, K;
for(I=0, J=N - 1, K=0; I<M && J>=0;)
{
    if (A[I]<=B[J])
        C[K++]=A[I++];
    else
        C[K++]=B[J--];
}
for (int T=I;T<M;T++)
    C[K++]=A[T];
for (T=J;T>=0;T--)
    C[K++]=B[T];
}

```

Q 4 Write a function in C++ which accepts an integer array and its size as arguments and assign the elements into a two dimensional array of integers in the following format

If the array is 1,2,3,4,5,6

if the array is 1,2,3

The resultant 2D array is

The resultant 2D array is

1 2 3 4 5 6

1 2 3

0 1 2 3 4 5

0 1 2

0 0 1 2 3 4

0 0 1

0 0 0 1 2 3

0 0 0 0 1 2

0 0 0 0 0 1

Ans: //Logic : Condition for putting the value is the position (i<=j) of 2D array otherwise put zero

```

void Change2Darray(int x[], int size)
{
for(i=0; i<size; i++)
{
    int k=0;
    for(int j=0; j< size; j++)
    {
        if(i<=j)
        {
            y[i][j]=x[k];
            k++;
        }
        else
            y[i][j]=0;
    }
}
for(i=0; i< size; i++)
{
    cout<<"\n";
for(int j=0; j< size; j++)
    cout<<y[i][j]<<" ";
}
}

```

Two Dimensional Array

In a two dimensional array the data is represented in form of rows and columns

Declaration of two dimensional array

Datatype nameofarray[no of rows][no of columns];

Eg int arr[5][3];

In the example given above a two dimensional array is declared with the name arr where there are 5 rows and three columns i.e total 15 (5X3) elements

Program for entering and displaying data(in matrix form) in two dimensional array

```

void main()
{
int arr[5][3],i,j;
cout<<"Enter array elements "<<endl;
for(i=0;i<5;i++)
{
    for(j=0;j<3;j++)
    {
        cin>>arr[i][j];
    }
}
cout<<"the array elements are"<<endl;
for(i=0;i<5;i++)
{
    for(j=0;j<3;j++)
    {

```



```

        cout<<arr[i][j]<<"\t";
    }
    cout<<endl;
}
getch();
}

```

Additional Question based on Two dimensional Array

Q1. Write a function in C++ that will accept a two D array and its row and column size as argument and find sum of rows and columns

Ans : void rowcolsum(int A[][],int N, int M)

```

{
    for (int i=0;i<N;i++)
    {
        int SumR=0;
        for (int j=0;j<M;j++)
            SumR+=A[i][j];
        cout<<SumR<<endl;
    }
    for (int i=0;i<N;i++)
    {
        int SumC=0;
        for (int j=0;j<M;j++)
            SumC+=A[j][i];
        cout<<SumC<<endl;
    }
}

```

Q2. Write a function in C++ to find the sum of both left and right diagonal elements from a two dimensional array (matrix).

Ans : void DiagSum(int A[][], int N)

```

{
    int SumD1=0,SumD2=0;
    for (int I=0;I<N;I++)
    {
        SumD1+=A[I][I];
        SumD2+=A[N-I-1][I];
    }
    cout<<"Sum of Diagonal 1:"<<SumD1<<endl;
    cout<<"Sum of Diagonal 2:"<<SumD2<<endl;
}

```

Address Calculation in Two Dimensional Array

Two dimensional array can be arranged in two manner

1. Row Major Order
2. Column Major Order

In Row Major Order the elements are shown Row Wise where as in Column Major Order the array elements are shown Column wise

Eg

If we have following two dimensional array

1	2	3
4	5	6
7	8	9

The array elements in **Row major Order** will be

1,2,3,4,5,6,7,8,9

The array elements in **Column Major Order** will be

1,4,7,2,5,8,3,6,9

To find the address of a particular row and column the formula in **Row Major Order** is

Address of $A[\text{row}][\text{column}] = B + w * (n(\text{row}) + \text{column})$

Where

B= Base address of the array

w= Word size

n= total no of columns in the array

To find the address of a particular row and column the formula in **Column Major Order** is

Address of $A[\text{row}][\text{column}] = B + w * (n(\text{Column}) + \text{row})$

Where

B= Base address of the array

w= Word size

n= total no of rows in the array

Q1. An array $x[30][10]$ is stored in the memory with each element requiring 4 bytes of storage. If the base address of x is 4500, find out memory locations of $x[12][8]$ if the content is stored along the row.

Ans: Here the array is stored in Row Major Order so

B=4500

W= 4

N= 10

As per the formula

Address of $A[\text{row}][\text{column}] = B + w * (n(\text{row}) + \text{column})$

=4500+4*(10(12)+8)

=4500+4*(128)

=4500+512

=5012

Q2. An array $x[30][10]$ is stored in the memory with each element requiring 4 bytes of storage. If the base address of x is 4500, find out memory locations of $x[12][8]$ if the content is stored along the row.

Ans: Here the array is stored in Column Major Order so

B=4500

W= 4

N= 30

As per the formula

Address of $A[\text{row}][\text{column}] = B + w * (n(\text{Column}) + \text{row})$

$$\begin{aligned}
&=4500+4*(30(8)+12) \\
&=4500+4*(252) \\
&=4500+1008 \\
&=5508
\end{aligned}$$

Q 3. An array P[20][30] is stored in the memory along the column with each of the element occupying 4 bytes, find out the Base Address of the array, if an element P[2][20] is stored at the memory location 5000.

Ans : Given, W=4, N=20, M=30, Loc(P[2][20])=5000

Column Major Formula:

$$\begin{aligned}
\text{Loc}(P[I][J]) &= \text{Base}(P) + W*(N*J+I) \\
\text{Loc}(P[2][20]) &= \text{Base}(P) + 4*(20*20+2) \\
\text{Base}(P) &= 5000 - 4*(400+2) \\
&= 5000 - 1608 \\
&= 3392
\end{aligned}$$

Q4. An array S[40][30] is stored in the memory along the row with each of the element occupying 2 bytes, find out the memory location for the element S[20][10], if an element S[15][5] is stored at the memory location 5500.

Ans. Given, W=2, N=40, M=30, Loc(S[15][5])=5500

Row Major Formula:

$$\begin{aligned}
\text{Loc}(S[I][J]) &= \text{Base}(S) + W*(M*I+J) \\
\text{Loc}(S[15][5]) &= \text{Base}(S) + 2*(30*15+5) \\
5500 &= \text{Base}(S) + 2*(450+5) \\
\text{Base}(S) &= 5500 - 910 = 4590
\end{aligned}$$

$$\begin{aligned}
\text{Loc}(S[20][10]) &= 4590 + 2*(30*20+10) \\
&= 4590 + 2*(600+10) \\
&= 4590 + 1220 = 5810
\end{aligned}$$

STACKS, QUEUES AND LINKED LIST

Stack

In computer science, a stack is a Last in, First out (LIFO) data structure. It simply means that an element that is inserted at the end will be deleted first. To Manage a stack all the insertion and deletion takes place from one position called "top".

One of the common uses of stack is in function call.

Operations on the Stack

There are two fundamental operations

Push

Pop

Push means to insert an element

Pop means to delete an element

Stack using array

```
#include<iostream.h>
const int size=5
class stack
{
```

```

int a[size]; //array a can store maximum 5 item of type int of the stack
int top; //top will point to the last item pushed onto the stack
public:
stack() //constructor to create an empty stack, top=-1 indicate that no item is present in the
{ // array
top=-1;
}
void push(int item)
{
if(top==size-1)
cout<<"stack is full, given item cannot be added";
else
a[++top]=item; //increment top by 1 then item at new position of the top in the array a
}
int pop()
{
if (top== -1)
{
cout<<"Stack is empty ";
return -1; //-1 indicates empty stack
}
else
return a[top--]; //return the item present at the top of the stack then decrement top by 1
};
};
void main()
{
stack s1;
s1.push(3);
s1.push(5);
cout<<s1.pop()<<endl;
cout<<s1.pop()<<endl;
cout<<s1.pop();
}

```

Queue

In computer science, a Queue is a First in, First out (FIFO) data structure. It simply means that an element that is inserted at the beginning will be deleted first. To Manage a queue all the insertion and deletion takes place from two different positions called "front" and "rear". Every element is inserted from the rear position and deleted from the front position in the queue.

```

Queue using array
#include<iostream.h>
const int size=5
class queue
{
int a[size]; //array a can store maximum 5 item of type int of the queue
int front,rear;

```

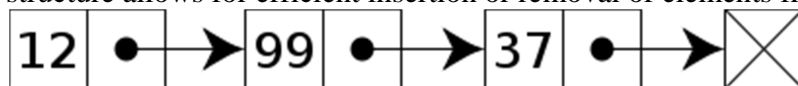
```

public:
queue() //constructor to create an empty queue
{
front=rear=-1;
}
void insert(int item)
{
If((front==0)&&(rear==size-1))
cout<<"queue is full, given item cannot be added";
else if((front==-1)||(rear==-1))
front=rear=0;
else
rear++;
a[rear]=item; }
int deletion()
{
if (front== -1)
{
cout<<"queue is empty ";
return -1;
}
else
return a[front++];
}
};
void main()
{
queue q1;
q1.insert(3);
q1.insert(5);
cout<<q1.deletion()<<endl;
cout<<q1.deletion()<<endl;
}

```

Linked List

A linked list is a data structure consisting of a group of nodes which together represent a sequence. Under the simplest form, each node is composed of a data and a reference (in other words, a *link*) to the next node in the sequence; more complex variants add additional links. This structure allows for efficient insertion or removal of elements from any position in the sequence.



Here in the figure is an example of a linked list whose nodes contain two fields: an integer value and a link to the next node. The last node is linked to a terminator used to signify the end of the list.

Linked lists are among the simplest and most common data structures. They can be used to implement several other common abstract data types, stacks, queues etc though it is not uncommon to implement the other data structures directly without using a list as the basis of implementation.

The principal benefit of a linked list over an array is that the list elements can easily be inserted or removed without reallocation or reorganization of the entire structure because the data items need not be stored contiguously in memory or on disk. Linked lists allow insertion and removal of nodes at any point in the list, and can do so with a constant number of operations if the link previous to the link being added or removed is maintained during list traversal. Linked list are dynamic structure where memory allocation takes place at run time.

Operation on a linked list

There are three basic operations on a linked list

Insertion

Deletion

Traversal

Inserting a node or element into Linked list :

Inserting an element into linked list contains 3 types .

1. Insertion at beginning of the Linked list
2. Insertion at the middle of the linked list
3. Insertion at the end of the linked list

1. Insertion at beginning of the Linked list :

An element can be inserted at the beginning of the Linked list by creating a new node at the beginning of the linked list and inserting an element into the node and assigning the address of the address of the next node or assigning the reference of the next node .

2. Insertion at the middle of the linked list:

An element can be inserted at the middle of the linked list. We need to know the data or element of the neighbouring element to insert an element at middle.

3. Insertion at the End of the linked list :

An element can be inserted at the end by simply traversing the node to the end and creating a new node at the end . Also the address pointer of the new node must be assigned to NULL

Deleting a node from the Linked list.

Deleting an element is similar to pop operation in Stack and Queue. A node can be deleted in 3 ways similar to Insertion.

1. Deleting a Node from the beginning of the Linked List
2. Deleting a node from the middle of the Linked list.
3. Deleting a node from the end of the Linked List .

Program of linked list

```
#include<iostream.h>
```

```
class node
```

```
{
```

```
public :
```

```
int d;
```

```
node *p;
```

```
};
```

```
class linked
```

```
{
```

```
public :
```

```

    node *start;
public :
    linked()
    {
        start=NULL;
    }
    void insert_at_beg();
    void insert_at_mid();
    void insert_at_end();
    int remove();
    void display();
};
void linked :: insert_at_end()
{
    node *temp;
    if(start==NULL)
    {
        start=new node [1];
        start->p=NULL;
        cout<<"\n Enter Data to insert \n";
        cin>>start->d;
    }
    else
    {
        temp=start;
        while(temp->p!=NULL)
            temp=temp->p;
        temp->p= new node [1];
        temp=temp->p;
        cout<<"\n Enter Data to insert \n";
        cin>>temp->d;
        temp->p=NULL;
    }
}
void linked :: insert_at_mid()
{
    node *temp,*next;
    int x;
    temp=start;
    cout<<"\n Enter the Neighbour element \n";
    cin>>x;
    while(temp->d!=x)
        temp=temp->p;
    next=temp->p;
    temp->p=new node [1];
    temp=temp->p;
    temp->p=next;
    cout<<"\n Enter data to insert \n";
    cin>>temp->d;
}

```

```

}
void linked :: insert_at_beg()
{
    node *temp;
    temp=new node [1];
    temp->p=start;
    start=temp;
    cout<<"\n Enter element to insert \n";
    cin>>temp->d;
}
int linked :: remove()
{
    node *temp,*prev;
    int x;
    if(start==NULL)
    {
        cout<<"\n Underflow -- No element to Delete \n";
        return 0;
    }
    cout<<"\n Enter element to delete \n";
    cin>>x;
    if(start->d==x && start->p==NULL)
    {
        delete start;
        start=NULL;
        cout<<"\n Deletion successfull \n";
        return 0;
    }
    if(start->d==x)
    {
        temp=start->p;
        delete start;
        start=temp;
        cout<<"\n Deletion Successfull \n";
        return 0;
    }
    temp=start;
    while(temp->d!=x)
    {
        prev=temp;
        temp=temp->p;
    }
    prev->p=temp->p;
    cout<<"\n Deletion Successfull \n";
    return x;
}
void linked :: display()
{
    node *temp;

```



```

temp=start;
cout<<"\n Linked List elements are \n";
while(temp->p!=NULL)
{
    cout<<" "<<temp->d<<" ";
    temp=temp->p;
}
cout<<temp->d<<endl;
}
int main()
{
    int n=0,a;
    linked l;
    do
    {
        cout<<"\n ***** M E N U *****\n";
        cout<<"\n1.Insert at beginning\n2.Insert at middle\n3.Insert at end\n4.Delete\n5.Display
elements";
        cout<<"\n6.Exit\n";
        cout<<"Enter option \n";
        cin>>a;
        switch(a)
        {
            case 1:
                l.insert_at_beg();
                break;
            case 2:
                l.insert_at_mid();
                break;
            case 3:
                l.insert_at_end();
                break;
            case 4:
                l.remove();
                break;
            case 5:
                l.display();
                break;
            case 6:
                n=1;
                break;
            default :
                cout<<"\n Invalid Entry \n";
                break;
        }
    }while(n!=1);
    return 0;
}

```

Implementation of stacks using a linked list

The stack which is implemented using linked list is called linked stack or dynamic stack

```
#include<iostream.h>
#include<conio.h>
struct node
{
int data;
node * next;
};
class stack
{
node *top;
public:
stack()
{
top=NULL;
}
void stackpush();
void stackpop();
void displaystack();
};
void stack::stackpush()
{
node *ptr;
ptr=new node;
cout<<"Enter the element to be pushed"<<endl;
cin>>ptr->data;
if(top==NULL)
ptr->next=NULL;
else
ptr->next=top;
top=ptr;
}
void stack ::stackpop()
{
node *ptr;
ptr=new node;
ptr=top;
cout<<"The popped element is "<<ptr->data;
top=top->next;
delete ptr;
}
void stack :: displaystack()
{
node *ptr;
ptr=new node;
ptr=top;
cout<<"The stack is "<<endl;
while(ptr!=NULL)
```

```

{
cout<<ptr->data<<endl;
ptr=ptr->next;
}
}
void main()
{
clrscr();
char ans;
stack s1;
do
{
s1.stackpush();
cout<<"wish to continue "<<endl;
cin>>ans;
}while(ans=='y');
s1.displaystack();
cout<<"Press any key to pop an element"<<endl;
getch();
s1.stackpop();
getch();
}

```

Implementation of queues using a linked list

The queue which is implemented using linked list is called linked queue or dynamic queue

```

#include<iostream.h>
#include<conio.h>

```

```

struct node
{
int data;
node * next;
};
class queue
{
node *front,*rear;
public:
queue()
{
rear=front=NULL;
}
void insqueue();
void delqueue();
void dispqueue();
};
void queue::insqueue()
{
node *ptr;
ptr=new node;
cout<<"Enter the element to be insert"<<endl;

```

```

cin>>ptr->data;
ptr->next=NULL;
if(rear==NULL)
    front=rear=ptr;
else
    {
    rear->next=ptr;
    rear=ptr;
    }
}
void queue ::delqueue()
{
node *ptr;
ptr=new node;
ptr=front;
cout<<"The deleted element is "<<ptr->data;
if(front==rear)
front=rear=NULL;
else
front=front->next;

delete ptr;
}

void queue :: dispqueue()
{
node *ptr;
ptr=new node;
ptr=front;
cout<<"The queue is "<<endl;
while(ptr!=NULL)
{
cout<<ptr->data<<endl;
ptr=ptr->next;
}
}
void main()
{
clrscr();
char ans;
queue q1;
do
{
q1.insqueue();
cout<<"wish to continue "<<endl;
cin>>ans;
}while(ans=='y');
q1.dispqueue();
cout<<"Press any key to delete an element ...."<<endl;

```

```

getch();
q1.delqueue();
getch();
}

```

Some question based on Board Examination Linked stack & Linked queue

Q1. Write a function in C++ to delete a node containing customer's information, from a dynamically allocated Queue of Customers implemented with the help of the following structure:

```

struct Customer
{
    int CNo;
    char CName[20];
    Customer *Link;
};

```

Ans: struct Customer

```

{
    int CNo;
    char CName[20];
    Customer *Link;
} *Front, *Rear, *ptr;
void DELETE()
{
    if(Front == NULL)
        cout<<"\n Queue Underflow\n";
    else
    {
        ptr = Front;
        Front = Front->Link;
        delete ptr;
    }
}

```

Q2. Write a function in C++ to delete a node containing Book's information, from a dynamically allocated Stack of Books implemented with the help of the following structure.

```

struct Book
{
    int BNo;
    char BName[20];
    Book *Next;
};

```

Ans: struct Book

```

{
    int BNo;
    char BName[20];
    Book *Next;
} *Front, *Rear, *ptr;
void POP()
{
    if(Front == NULL)
        cout<<"\n Stack Underflow\n";
}

```

```

else
{
    ptr = Front;
    Front = Front→Link;
    delete ptr;    }
}

```

Application of Stack

Q 1. Evaluate the postfix notation of expression.

4, 10, 5, +, *, 15, 3, /, -

Sno.	Symbol	Stack
0		[
1	4	[4
2	10	[4,10
3	5	[4,10,5
4	+	[4 [4,15
5	*	[[60
6	15	[60,15
7	3	[60,15,3
8	/	[60 [60,5
9	-	[[55
10]	55 Ans

Q2 Evaluate the following postfix expression:

5,20,15,-,*,25,2,*,+

Sno.	Symbol	Stack
0		[
1	5	[5
2	20	[5,20
3	15	[5,20,15
4	-	[5 [5,5
5	*	[[25
6	25	[25,25
7	2	[25,25,2
8	*	[25 [25,50
9	+	[[75
10]	75 Ans

UNIT -3

DATABASES MANAGEMENT SYSTEM AND SQL

Basic Database concepts

Data :- Raw facts and figures which are useful to an organization. We cannot take decisions on the basis of data.

Information:- Well processed data is called information. We can take decisions on the basis of information

Field: Set of characters that represents specific data element.

Record: Collection of fields is called a record. A record can have fields of different data types.

File: Collection of similar types of records is called a file.

Table: Collection of rows and columns that contains useful data/information is called a table. A table generally refers to the passive entity which is kept in secondary storage device.

Relation: Relation (collection of rows and columns) generally refers to an active entity on which we can perform various operations.

Database: Collection of logically related data along with its description is termed as database.

Tuple: A row in a relation is called a tuple.

Attribute: A column in a relation is called an attribute. It is also termed as field or data item.

Degree: Number of attributes in a relation is called degree of a relation.

Cardinality: Number of tuples in a relation is called cardinality of a relation.

Primary Key: Primary key is a key that can uniquely identifies the records/tuples in a relation. This key can never be duplicated and NULL.

Foreign Key: Foreign Key is a key that is defined as a primary key in some other relation. This key is used to enforce referential integrity in RDBMS.

Candidate Key: Set of all attributes which can serve as a primary key in a relation.

Alternate Key: All the candidate keys other than the primary keys of a relation are alternate keys for a relation.

DBA: Data Base Administrator is a person (manager) that is responsible for defining the data base schema, setting security features in database, ensuring proper functioning of the data bases etc.

Select Operation:The select operation selects tuples from a relation which satisfy a given condition. It is denoted by lowercase Greek Letter σ (sigma).

Project Operation:The project operation selects columns from a relation which satisfy a given condition. It is denoted by lowercase Greek Letter π (pi). It can be thought of as picking a sub set of all available columns.

Union Operation: The union (denoted as \cup) of a collection of relations is the set of all distinct tuples in the collection. It is a binary operation that needs two relations.

Set Difference Operation: This is denoted by $-$ (minus) and is a binary operation. It results in a set of tuples that are in one relation but not in another

Structured Query Language

SQL is a non procedural language that is used to create, manipulate and process the databases(relations).

Characteristics of SQL

2. It is very easy to learn and use.
3. Large volume of databases can be handled quite easily.
4. It is non procedural language. It means that we do not need to specify the procedures to

accomplish a task but just to give a command to perform the activity.

5. SQL can be linked to most of other high level languages that makes it first choice for the database programmers.

Processing Capabilities of SQL

The following are the processing capabilities of SQL

1. Data Definition Language (DDL)

DDL contains commands that are used to create the tables, databases, indexes, views, sequences and synonyms etc.

e.g: Create table, create view, create index, alter table etc.

2. Data Manipulation Language (DML)

DML contains command that can be used to manipulate the data base objects and to query the databases for information retrieval.

e.g Select, Insert, Delete, Update etc.

3. View Definition:

DDL contains set of command to create a view of a relation.

e.g : create view

4. Data Control Language:

This language is used for controlling the access to the data. Various commands like GRANT, REVOKE etc are available in DCL.

5. Transaction Control Language (TCL)

TCL include commands to control the transactions in a data base system. The commonly used commands in TCL are COMMIT, ROLLBACK etc.

Operators in SQL:

The following are the commonly used operators in SQL

1. Arithmetic Operators +, -, *, /
2. Relational Operators =, <, >, <=, >=, <>
3. Logical Operators OR, AND, NOT

Data types of SQL

Just like any other programming language, the facility of defining data of various types is available in SQL also. Following are the most common data types of SQL.

- 1) NUMBER e.g. Number(n,d) Number (5,2)
- 2) CHAR CAHR(SIZE)
- 3) VARCHAR / VARCHAR2 VARCHAR2(SIZE)
- 4) DATE DD-MON-YYYY
- 5) LONG
- 6) RAW/LONG RAW

Constraints:

Constraints are the conditions that can be enforced on the attributes of a relation. The constraints come in play when ever we try to insert, delete or update a record in a relation.

.Not null ensures that we cannot leave a column as null. That is a value has to be supplied for that column.

e.g name varchar(25) not null;

Unique constraint means that the values under that column are always unique.

e.g Roll_no number(3) unique;

Primary key constraint means that a column can not have duplicate values and not even a null value.

e.g.Roll_no number(3) primary key;

The main difference between unique and primary key constraint is that a column specified as unique may have null value but primary key constraint does not allow null values in the column.

Foreign key is used to enforce referential integrity and is declared as a primary key in some other table.

e.g `cust_id varchar(5) references master(cust_id);`

it declares `cust_id` column as a foreign key that refers to `cust_id` field of table `master`. That means we cannot insert that value in `cust_id` field whose corresponding value is not present in `cust_id` field of `master` table.

Check constraint limits the values that can be inserted into a column of a table.

e.g `marks number(3) check(marks >= 0);`

The above statement declares `marks` to be of type `number` and while inserting or updating the value in `marks` it is ensured that its value is always greater than or equal to zero.

Default constraint is used to specify a default value to a column of a table automatically.

This default value will be used when user does not enter any value for that column.

e.g `balance number(5) default = 0;`

SYNTAX of some SQL COMMANDS :

1. `SELECT column list FROM <table name> WHERE <condition> GROUP BY <column name(s)> HAVING <search condition> ORDER BY column name;`
2. `INSERT INTO <table name> [<column list>] VALUES (<value>, <value>, ...);`
3. `DELETE FROM <table name> [WHERE <predicate>];`
4. `UPDATE <table name> SET <column name> = <new value> [WHERE <predicate>];`
5. `CREATE TABLE <table name> (<column name><data type> [(size)] <column constraint>, <column name><data type> [(size)] <column constraint>, ... <table constraint> (<column name> [<column name> ...] ...);`
6. `CREATE VIEW <view name> [(<column name>, <column name>,...)] AS <SELECT command>;`
7. `ALTER TABLE <table name> ADD / MODIFY <column name><data type><size>;`
8. `DROP TABLE <table name>;`
9. `DROP VIEW <view name>;`

Consider the following Table :-

Roll_ no	Name	Class	Marks	City
101	Rohan	XI	400	Jammu
102	Aneeta Chopra	XII	390	Udhampur
103	Pawan Kumar	IX	298	Amritsar
104	Rohan	IX	376	Jammu
105	Sanjay	VII	240	Gurdaspur
113	Anju Mahajan	VIII	432	Pathankot

Eliminating Duplicate/Redundant data

DISTINCT keyword is used to restrict the duplicate rows from the results of a `SELECT` statement.

e.g. `SELECT DISTINCT name FROM student;`

The above command returns

Name

Rohan

Aneeta Chopra

Pawan Kumar

Conditions based on a range

SQL provides a BETWEEN operator that defines a range of values that the column value must fall for the condition to become true.

e.g. SELECT Roll_no, name FROM student WHERE Roll_no BETWEEN 100 AND 103;

The above command displays Roll_no and name of those students whose Roll_no lies in the range 100 to 103 (both 100 and 103 are included in the range).

Conditions based on a list

To specify a list of values, IN operator is used. This operator select values that match any value in the given list.

e.g. SELECT * FROM student WHERE city IN ('Jammu', 'Amritsar', 'Gurdaspur');

The above command displays all those records whose city is either Jammu or Amritsar or Gurdaspur.

Conditions based on Pattern

SQL provides two wild card characters that are used while comparing the strings with LIKE operator.

a. percent(%) Matches any string

b. Underscore(_) Matches any one character

e.g. SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "%3";

displays those records where last digit of Roll_no is 3 and may have any number of characters in front.

e.g. SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "1_3";

displays those records whose Roll_no starts with 1 and second letter may be any letter but ends with digit 3.

ORDER BY Clause

ORDER BY clause is used to display the result of a query in a specific order(sorted order).

The sorting can be done in ascending or in descending order. It should be kept in mind that the actual data in the database is not sorted but only the results of the query are displayed in sorted order.

e.g. SELECT name, city FROM student ORDER BY name;

The above query returns name and city columns of table student sorted by name in increasing/ascending order.

e.g. SELECT * FROM student ORDER BY city DESC;

It displays all the records of table student ordered by city in descending order.

Note:- If order is not specified that by default the sorting will be performed in ascending order.

GROUP BY Clause

The GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

The syntax for the GROUP BY clause is:

SELECT column1, column2, ... column_n, aggregate_function (expression)

FROM tables

WHERE conditions

GROUP BY column1, column2, ... column_n;

aggregate_function can be a function such as SUM, COUNT, MAX, MIN, AVG etc.

```
e.g      SELECT name, COUNT(*) as "Number of employees"
FROM student
WHERE marks>350
GROUP BY city;
```

The INSERT Command:

The tuples are added to relation using INSERT command of SQL.

Syntax:

```
INSERT INTO <table-name>[<column list>]
VALUES (<value>,<value>,<value>,...);
```

Example :

Enter a new record in student table

```
INSERT INTO student (sid,sname,fname,age,class,address);
VALUES(101,"Mohan","Pawan",15,"8","Jaipur");
```

sid sname fname age class address

```
101 Mohan Pawan 15 8 Jaipur
```

The DELETE Command:

The delete command removes the tuples from the tables. This command remove the entire row from the table and not the individual field. So no filed argument is needed.

Syntax

```
DELETE FROM <table-name>
WHERE <condition>;
```

Example

Delete all the records of employee whose salary is less than 3000

```
DELETE FROM emp
WHERE sal<3000;
```

To delete all the record from the table:

```
DELET FROM<table-name>;
```

The UPDATE command is used to changes some values in existing rows. The UPDATE command specifies the rows to be changed using the WHERE clause, and new data using the SET keyword.

Example:

Update the salary of employee to 5000 whose employee code is 1011.

```
UPDATE emp
SET sal=5000
WHERE empno=1011;
```

The ALTER TABLE Command:

The ALTER command is used to change the definition of existing table.

a)It can be used to add columns to a table.

Syntax (to add a column to a table):

```
ALTER TABLE <table-name> ADD <column-name>
<data type> <size>;
```

b)To modify existing columns of a table:

Syntax:

```
ALTER TABLE <table-name>
MODIFY (Columnname newdatatype (news�));
```

Example:

To modify column job of table emp to have new width of 30 character

```
ALTER TABLE emp
```

MODIFY (job char(30));

The DROP Command

The DROP command is used to drop the table from the database. For dropping a table all the tuples should be deleted first i.e the table should be empty.

Syntax:

DROP TABLE <table-name>

Solved Exercise :

Q1. Consider the following tables ACTIVITY and COACH. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).

Table: ACTIVITY

ACode	ActivityName	ParticipantsNum	PrizeMoney	ScheduleDate
1001	Relay 100x4	16	10000	23-Jan-2004
1002	High jump	10	12000	12-Dec-2003
1003	Shot Put	12	8000	14-Feb-2004
1005	Long Jump	12	9000	01-Jan-2004
1008	Discuss Throw	10	15000	19-Mar-2004

Table: COACH

PCode	Name	ACode
1	Ahmad Hussain	1001
2	Ravinder	1008
3	Janila	1001
4	Naaz	1003

- (i) To display the name of all activities with their ACodes in descending order.
- (ii) To display sum of PrizeMoney for each of the Number of participants groupings (as shown in column ParticipantsNum 10,12,16).
- (iii) To display the coach's name and ACodes in ascending order of ACode from the table COACH.
- (iv) To display the content of the GAMES table whose ScheduleDate earlier than 01/01/2004 in ascending order of ParticipantNum.
- (v) SELECT COUNT(DISTINCT ParticipantsNum) FROM ACTIVITY;
- (vi) SELECT MAX(ScheduleDate),MIN(ScheduleDate) FROM ACTIVITY;
- (vii) SELECT SUM(PrizeMoney) FROM ACTIVITY;
- (viii) SELECT DISTINCT ParticipantNum FROM COACH;

- Ans : (i) SELECT ActivityName, ACode FROM ACTIVITY ORDER BY Acode DESC;
- (ii) SELECT SUM(PrizeMoney),ParticipantsNum FROM ACTIVITY GROUP BY ParticipantsNum;
- (iii) SELECT Name, ACode FROM COACH ORDER BY ACode;
- (iv) SELECT * FROM ACTIVITY WHERE ScheduleDate<'01-Jan-2004' ORDER BY ParticipantsNum;
- (v) 3
- (vi) 19-Mar-2004 12-Dec-2003
- (vii) 54000
- (viii) 16

10
12

Q2. Consider the following tables GAMES and PLAYER. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).

Table: GAMES

GCode	GameName	Number	PrizeMoney	ScheduleDate
101	Carom Board	2	5000	23-Jan-2004
102	Badminton	2	12000	12-Dec-2003
103	Table Tennis	4	8000	14-Feb-2004
105	Chess	2	9000	01-Jan-2004
108	Lawn Tennis	4	25000	19-Mar-2004

Table: PLAYER

PCode	Name	Gcode
1	Nabi Ahmad	101
2	Ravi Sahai	108
3	Jatin	101
4	Nazneen	103

- (i) To display the name of all Games with their Gcodes.
- (ii) To display details of those games which are having PrizeMoney more than 7000.
- (iii) To display the content of the GAMES table in ascending order of ScheduleDate.
- (iv) To display sum of PrizeMoney for each of the Number of participation groupings (as shown in column Number 2 or 4).
- (v) SELECT COUNT(DISTINCT Number) FROM GAMES;
- (vi) SELECT MAX(ScheduleDate),MIN(ScheduleDate) FROM GAMES;
- (vii) SELECT SUM(PrizeMoney) FROM GAMES;
- (viii) SELECT DISTINCT Gcode FROM PLAYER;

- Ans : (i) SELECT GameName,Gcode FROM GAMES;
(ii) SELECT * FROM GAMES WHERE PrizeMoney>7000;
(iii) SELECT * FROM GAMES ORDER BY ScheduleDate;
(iv) SELECT SUM(PrizeMoney),Number FROM GAMES GROUP BY Number;
(v) 2
(vi) 19-Mar-2004 12-Dec-2003
(vii) 59000
(viii) 101
103
108

Q3. Consider the following tables HOSPITAL. Give outputs for SQL queries (i) to (iv) and write SQL commands for the statements (v) to (viii).

No	Name	Age	Department	Dateofadmin	Charge	Sex
1	Arpit	62	Surgery	21/01/06	300	M
2	Zayana	18	ENT	12/12/05	250	F
3	Kareem	68	Orthopedic	19/02/06	450	M
4	Abhilash	26	Surgery	24/11/06	300	M
5	Dhanya	24	ENT	20/10/06	350	F

6	Siju	23	Cardiology	10/10/06	800	M
7	Ankita	16	ENT	13/04/06	100	F
8	Divya	20	Cardiology	10/11/06	500	F
9	Nidhin	25	Orthopedic	12/05/06	700	M
10	Hari	28	Surgery	19/03/06	450	M

- (i) Select SUM(Charge) from HOSPITAL where Sex='F';
- (ii) Select COUNT(DISTINCT Department) from HOSPITAL;
- (iii) Select SUM(Charge) from HOSPITAL group by Department;
- (iv) Select Name from HOSPITAL where Sex='F' AND Age > 20;
- (v) To show all information about the patients whose names are having four characters only.
- (vi) To reduce Rs 200 from the charge of female patients who are in Cardiology department.
- (vii) To insert a new row in the above table with the following data :
11, 'Rakesh', 45, 'ENT', {08/08/08}, 1200, 'M'
- (viii) To remove the rows from the above table where age of the patient > 60.

Ans : (i) 1200

(ii) 4

(iii) 1050

700

1150

1300

(iv) Dhanya

(v) SELECT * FROM HOSPITAL WHERE NAME LIKE “_ _ _ _”;

(vi) UPDATE HOSPITAL SET CHARGE = CHARGE – 200 WHERE (DEPARTMENT = 'CARDIOLOGY' AND SEX = 'F');

(vii) INSERT INTO HOSPITAL VALUES(11, 'Rakesh', 45, 'ENT', {08/08/08}, 1200, 'M');

(viii) DELETE FROM HOSPITAL WHERE AGE > 60;

Q4. Consider the following tables BOOKS. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).

Table : BOOKS

B_Id	Book_Name	Author_Name	Publisher	Price	Type	Quantity
C01	Fast Cook	Lata Kapoor	EPB	355	Cookery	5
F01	The Tears	William Hopkins	First	650	Fiction	20
T01	My C++	Brain & Brooke	FPB	350	Text	10
T02	C++ Brain	A.W.Rossaine	TDH	350	Text	15
F02	Thunderbolts	Anna Roberts	First	750	Fiction	50

- i). To list the names from books of Text type.
- ii). To display the names and price from books in ascending order of their price.
- iii). To increase the price of all books of EPB publishers by 50.
- iv). To display the Book_Name, Quantity and Price for all C++ books.
- v). Select max(price) from books;
- vi). Select count(DISTINCT Publishers) from books where Price >=400;
- vii). Select Book_Name, Author_Name from books where Publishers = 'First';
- viii). Select min(Price) from books where type = 'Text';

- Ans : (i) SELECT Book_Name FROM BOOKS WHERE Type = 'Text';
(ii) SELECT Book_Name, Price FROM BOOKS ORDER BY Price;
(iii) UPDATE BOOKS SET Price = Price + 50 WHERE Publisher = 'EPB';
(iv) SELECT Book_Name, Quantity, Price FROM BOOKS WHERE Book_Name LIKE '%C++%';
(v) 750
(vi) 2
(vii) The Tears William Hopkins
Thunderbolts Anna Roberts
(viii) 350

Q5. Consider the tables ITEMS & COMPANY. Write SQL commands for the statements (i) to (iv) and give the outputs for SQL queries (v) to (viii).

Table : ITEMS

ID	PNAME	PRICE	MDATE	QTY
T001	Soap	12.00	11/03/2007	200
T002	Paste	39.50	23/12/2006	55
T003	Deodorant	125.00	12/06/2007	46
T004	Hair Oil	28.75	25/09/2007	325
T005	Cold Cream	66.00	09/10/2007	144
T006	Tooth Brush	25.00	17/02/2006	455

Table : COMPANY

ID	COMP	City
T001	HLL	Mumbai
T008	Colgate	Delhi
T003	HLL	Mumbai
T004	Paras	Haryana
T009	Ponds	Noida
T006	Wipro	Ahmedabad

- i).** To display PNAME, PRICE * QTY only for the city Mumbai.
ii). To display product name, company name & price for those items which IDs are equal to the IDs of company.
iii). To delete the items produced before 2007.
iv). To increase the quantity by 20 for soap and paste.
v). SELECT COUNT(*) FROM ITEMS WHERE ITEMS.ID=COMPANY.ID;
vi). SELECT PNAME FROM ITEMS WHERE PRICE=SELECT MIN(PRICE) FROM ITEMS;
vii). SELECT COUNT(*) FROM COMPANY WHERE COMP LIKE "P_ _ _ _";
viii). SELECT PNAME FROM ITEMS WHERE QTY<100;

- Ans : (i) SELECT PNAME, QTY*PRICE FROM ITEMS WHERE ITEMS.ID = COMPANY.ID AND COMPANY.City='Mumbai';
(ii) SELECT PNAME, COMP, PRICE FROM ITEMS, COMPANY WHERE ITEMS.ID = COMPANY.ID
(iii) DELETE FROM ITEMS WHERE MDATE < {01/01/2007};

(iv) UPDATE ITEMS SET QTY = QTY + 20 WHERE PNAME = 'Soap' OR PNAME = 'Paste';

(v) 4

(vi) Soap

(vii) 2

(viii) Paste Deodorant

Unsolved Problems :

Q-1 Write the SQL query commands based on following table

Table : Book

Book_id	Book_name	Author_name	Publisher	Price	Type	Quantity
C0001	Fast Cook	Lata Kapoor	EPB	355	Cookery	5
F0001	The Tears	William Hopkins	First Publi.	650	Fiction	20
T0001	My First c++	Brain & Brooke	FPB	350	Text	10
T0002	C++ Brain works	A.W. Rossaine	TDH	350	Text	15
F0002	Thunderbolts	Anna Roberts	First Publ.	750	Fiction	50

Table : issued

Book Id	Quantity Issued
T0001	4
C0001	5
F0001	2

Write SQL query for (a) to (f)

(a) To show book name, Author name and price of books of First Pub. Publisher

(b) To list the names from books of text type

(c) To Display the names and price from books in ascending order of their prices.

(d) To increase the price of all books of EPB publishers by 50.

(e) To display the Book_Id, Book_name and quantity issued for all books which have been issued

(f) To insert a new row in the table issued having the following data. „F0003“, 1

(g) Give the output of the following

i. Select Count(*) from Books

ii. Select Max(Price) from books where quantity >=15

iii. Select book_name, author_name from books where publishers="first publ."

iv. Select count(distinct publishers) from books where Price >=400

Q-2 Write the SQL query commands based on following table

Table : SchoolBus

Rtno	Area_oved	Capacity	Noofstudents	Distance	Transporter	Charges
1	Vasant kunj	100	120	10	Shivamtravels	100000
2	Hauz Khas	80	80	10	Anand travels	85000
3	Pitampura	60	55	30	Anand travels	60000
4	Rohini	100	90	35	Anand travels	100000
5	Yamuna Vihar	50	60	20	Bhalla Co.	55000

6	Krishna Nagar	70	80	30	Yadav Co.	80000
7	Vasundhara	100	110	20	Yadav Co.	100000
8	Paschim Vihar	40	40	20	Speed travels	55000
9	Saket	120	120	10	Speed travels	100000
10	Jank Puri	100	100	20	Kisan Tours	95000

- (b) To show all information of students where capacity is more than the no of student in order of rtno.
- (c) To show area_covered for buses covering more than 20 km., but charges less then 80000.
- (d) To show transporter wise total no. of students traveling.
- (e) To show rtno, area_covered and average cost per student for all routes where average cost per student is - charges/noofstudents.
- (f) Add a new record with following data:
 (11, "Moti bagh",35,32,10," kisan tours ", 35000)
- (g) Give the output considering the original relation as given:
- (i) select sum(distance) from schoolbus where transporter= "Yadav travels";
- (ii) select min(noofstudents) from schoolbus;
- (iii) select avg(charges) from schoolbus where transporter= "Anand travels";
- (i) select distinct transporter from schoolbus;

Q-3 Write the SQL query commands based on following table

TABLE : GRADUATE

S.NO	NAME	STIPEND	SUBJECT	AVERAGE	DIV.
1	KARAN	400	PHYSICS	68	I
2	DIWAKAR	450	COMP. Sc.	68	I
3	DIVYA	300	CHEMISTRY	62	I
4	REKHA	350	PHYSICS	63	I
5	ARJUN	500	MATHS	70	I
6	SABINA	400	CEHMISTRY	55	II
7	JOHN	250	PHYSICS	64	I
8	ROBERT	450	MATHS	68	I
9	RUBINA	500	COMP. Sc.	62	I
10	VIKAS	400	MATHS	57	II

- (a) List the names of those students who have obtained DIV 1 sorted by NAME.
- (b) Display a report, listing NAME, STIPEND, SUBJECT and amount of stipend received in a year assuming that the STIPEND is paid every month.
- (c) To count the number of students who are either PHYSICS or COMPUTER SC graduates.
- (d) To insert a new row in the GRADUATE table: 11,"KAJOL", 300, "computer sc", 75, 1
- (e) Give the output of following sql statement based on table GRADUATE:
- (i) Select MIN(AVERAGE) from GRADUATE where SUBJECT="PHYSICS";
- (ii) Select SUM(STIPEND) from GRADUATE WHERE div=2;
- (iii) Select AVG(STIPEND) from GRADUATE where AVERAGE>=65;
- (iv) Select COUNT(distinct SUBDJECT) from GRADUATE;
- (f) Assume that there is one more table GUIDE in the database as shown below:

Table: GUIDE

MAINAREA	ADVISOR
PHYSICS	VINOD
COMPUTER SC	ALOK
CHEMISTRY	RAJAN
MATHEMATICS	MAHESH

g) What will be the output of the following query:

SELECT NAME, ADVISOR FROM GRADUATE, GUIDE WHERE SUBJECT= MAINAREA;

Q-4 Write SQL command for (i) to (vii) on the basis of the table SPORTS

Table: SPORTS

Student NO	Class	Name	Game1	Grade	Game2	Grade2
10	7	Sammer	Cricket	B	Swimming	A
11	8	Sujit	Tennis	A	Skating	C
12	7	Kamal	Swimming	B	Football	B
13	7	Venna	Tennis	C	Tennis	A
14	9	Archana	Basketball	A	Cricket	A
15	10	Arpit	Cricket	A	Atheletics	C

- Display the names of the students who have grade 'C' in either Game1 or Game2 or both.
- Display the number of students getting grade 'A' in Cricket.
- Display the names of the students who have same game for both Game1 and Game2.
- Display the games taken up by the students, whose name starts with 'A'.
- Assign a value 200 for Marks for all those who are getting grade 'B' or grade 'A' in both Game1 and Game2.
- Arrange the whole table in the alphabetical order of Name.
- Add a new column named 'Marks'.

Q-5 Write SQL command for (i) to (vii) on the basis of the table Employees & EmpSalary

Table: Employees

Empid	Firstname	Lastname	Address	City
010	Ravi	Kumar	Raj nagar	GZB
105	Harry	Waltor	Gandhi nagar	GZB
152	Sam	Tones	33 Elm St.	Paris
215	Sarah	Ackerman	440 U.S. 110	Upton
244	Manila	Sengupta	24 Friends street	New Delhi
300	Robert	Samuel	9 Fifth Cross	Washington
335	Ritu	Tondon	Shastri Nagar	GZB
400	Rachel	Lee	121 Harrison St.	New York
441	Peter	Thompson	11 Red Road	Paris

Table: EmpSalary

Empid	Salary	Benefits	Designation
010	75000	15000	Manager
105	65000	15000	Manager
152	80000	25000	Director
215	75000	12500	Manager
244	50000	12000	Clerk
300	45000	10000	Clerk
335	40000	10000	Clerk
400	32000	7500	Salesman
441	28000	7500	salesman

Write the SQL commands for the following :

- (i) To show firstname,lastname,address and city of all employees living in paris
- (ii) To display the content of Employees table in descending order of Firstname.
- (iii) To display the firstname,lastname and total salary of all managers from the tables Employee and empsalary , where total salary is calculated as salary+benefits.
- (iv) To display the maximum salary among managers and clerks from the table Empsalary.
- (v) Give the Output of following SQL commands:
 - a) Select firstname,salary from employees ,empsalary where designation = 'Salesman' and Employees.empid=Empsalary.empid;
 - b) Select count(distinct designation) from empsalary;
 - c) Select designation, sum(salary) from empsalary group by designation having count(*) >2;
 - d) Select sum(benefits) from empsalary where designation ='Clerk';

UNIT-4

Boolean Algebra

Truth table:

Truth table is a table, which represents all the possible values of logical variables/statements along with all the possible results of given combinations of values.

Logical Operators:

Logical operators are derived from the Boolean algebra, which is the mathematical representation of the concepts without going into the meaning of the concepts.

1. NOT Operator—Operates on single variable. It gives the complement value of variable.

X	$\overline{\text{X}}$
0	1
1	0

2. OR Operator -It is a binary operator and denotes logical Addition operation and is represented by "+" symbol

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

3. AND Operator – AND Operator performs logical multiplications and symbol is (.) dot.

Truth table:

X	Y	X.Y
0	0	0
0	1	0
1	0	0
1	1	1

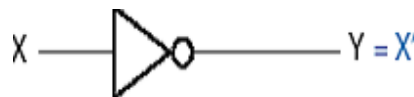
Basic Logic Gates

A gate is simply an electronic circuit, which operates on one or more signals to produce an output signal. Gates are digital circuits because the input and output signals are either low (0) or high (1). Gates also called logic circuits.

There are three types of logic gates:

1. Inverter (NOT gate)
2. OR gate
3. AND gate

1. NOT gate: This gate takes one input and gives a single output. The symbol of this logic gate is



This circuit is used to obtain the compliment of a value.

If X = 0, then X' = 1.

The truth table for NOT gate is :

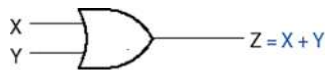
X	\overline{X}
0	1
1	0

2. OR gate : The OR gate has two or more input signals but only one output signal if any of the input signal is 1(high) the output signal is 1(high).

Truth Table for Two Input OR gate is :

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

The circuit diagram of two inputs OR gate is:-



AND gate The AND gate have two or more than two input signals and produce an output signal. When all the inputs are 1(High) then the output is 1 otherwise output is 0 only.

X	Y	F=X.Y
0	0	0
0	1	0
1	0	0
1	1	1

Circuit diagram of Two input AND gate



Basic postulates of Boolean Algebra:

Principal of Duality

This principal states that we can derive a Boolean relation from another Boolean relation by performing simple steps. The steps are:-

1. Change each AND(.) with an OR(+) sign
2. Change each OR(+) with an AND(.) sign
3. Replace each 0 with 1 and each 1 with 0

e.g

$$0+0=0 \quad \text{then dual is} \quad 1 \cdot 1=1$$

$$1+0=1 \quad \text{then dual is} \quad 0 \cdot 1=0$$

Basic theorem of Boolean algebra

Basic postulates of Boolean algebra are used to define basic theorems of Boolean algebra that provides all the tools necessary for manipulating Boolean expression.

1. Properties of 0 and 1

- (a) $0 + X = X$
 - (b) $1 + X = 1$
 - (c) $0 \cdot X = 0$
 - (d) $1 \cdot X = X$
2. Idempotence Law

- (a) $X + X = X$
- (b) $X \cdot X = X$

3. Involution Law

$$\overline{\overline{X}} = X$$

4. Complementarity Law

(a) $X + X = \overline{1}$

(b) $X \cdot X = \overline{0}$

5. Commutative Law

(a) $X + Y = Y + X$

(b) $X \cdot Y = Y \cdot X$

6. Associative Law

(a) $X + (Y + Z) = (X + Y) + Z$

(b) $X(YZ) = (XY)Z$

7. Distributive Law

(a) $X(Y + Z) = XY + XZ$

(b) $X = YZ = (X + Y)(X + Z)$

8. Absorption Law

(a) $X + XY = X$

(b) $X(X + Y) = X$

Some other rules of Boolean algebra

$$\overline{\overline{X}} + XY = X + Y$$

Demorgan's Theorem

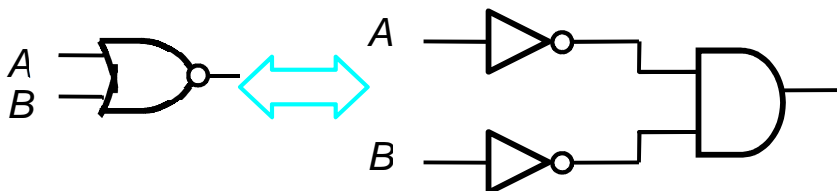
A mathematician named DeMorgan developed a pair of important rules regarding group complementation in Boolean algebra.

Demorgan's First Theorem:

This rule states that the compliment of OR of two operands is same as the AND of the compliments of those operands.

Mathematically it can be written as:-

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

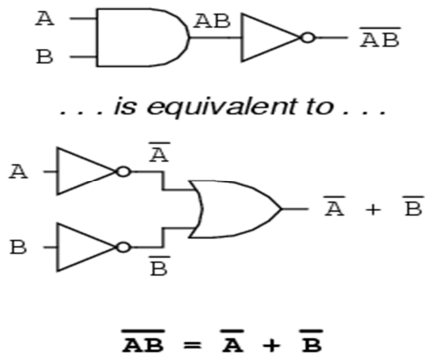


Demorgan's Second Theorem:

This rule states that the compliment of AND of two operands is same as the OR of the compliments of those operands.

Mathematically it can be written as:-

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$



Algebraic proof of DE Morgan's Law

Proof: We show that $a+b$ and $a'b'$ are complementary.

In other words, we show that both of the following are true (P4):

$$(a+b) + (a'b') = 1, (a+b)(a'b') = 0.$$

First part

$$\begin{aligned} &(a+b)+(a'b') \\ &=(a+b+a')(a+b+b') \quad (\text{Distribution Law}) \\ &=(1+b)(a+1) \quad (\text{Complement law}) \\ &=1 \end{aligned}$$

Second part :- $(a+b)(a'b')$

$$\begin{aligned} &=(a'b')(a+b) \quad (\text{Commutative law}) \\ &=a'b'a+a'b'b \quad (\text{Distribution Law}) \\ &=0*b'+a*0 \quad (x*0=0) \\ &=0+0 \\ &=0 \end{aligned}$$

Derivation of Boolean expression:-

Minterms and Maxterms

- Consider two binary variables x and y combined with an AND operation.
 $x'y', x'y, xy', xy$
Each of these four AND terms represents one of the distinct areas in the Venn diagram and is called a *minterm* or *standard product*.
- Consider two binary variables x and y combined with an OR operation.
 $x'+y', x'+y, x+y', x+y$
Each of these four OR terms represents one of the distinct areas in the Venn diagram and is called a *maxterm* or *standard sum*.
- n Variables can be combined to form 2^n minterms or maxterms.

Minterms and Maxterms for Three Binary Variables						
			Minterms		Maxterms	
x	y	Z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x+y+z$	M_0
0	0	1	$x'y'z$	m_1	$x+y+z'$	M_1
0	1	0	$x'yz'$	m_2	$x+y'+z$	M_2
0	1	1	$x'yz$	m_3	$x+y'+z'$	M_3
1	0	0	$xy'z'$	m_4	$x'+y+z$	M_4
1	0	1	$xy'z$	m_5	$x'+y+z'$	M_5
1	1	0	xyz'	m_6	$x'+y'+z$	M_6
1	1	1	xyz	m_7	$x'+y'+z'$	M_7

- A Boolean function may be represented algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those terms.

Functions of Three Variables				
X	y	z	Function F1	Function F2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$F1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

$$F2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

- The complement of a Boolean function may be read from the truth table by forming a minterm for each combination that produces a 0 in the function and then ORing those terms.

$$F1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

Example: Express the Boolean function $F(A,B,C) = AB + C$ as a sum of minterms.

Step 1 – Each term must contain all variables

$$AB = AB(C + C') = ABC + ABC'$$

$$C = C(A + A') = AC + A'C$$

$$= AC(B + B') + A'C(B + B')$$

$$= ABC + AB'C + A'BC + A'B'C$$

Step 2 – OR all new terms, eliminating duplicates

$$F(A,B,C) = A'B'C + A'BC + AB'C + ABC' + ABC$$

$$= m_1 + m_3 + m_5 + m_6 + m_7$$

$$= \Sigma(1, 3, 5, 6, 7)$$

Example: Express the Boolean function $F(x,y,z) = x'y + xz$ as a product of maxterms.

Step 1 – Convert the function into OR terms using the distributive law

$$F(x,y,z) = (x'y + x)(x'y + z)$$

$$= (x + x')(y + x)(x' + z)(y + z)$$

$$= (y + x)(x' + z)(y + z)$$

Step 2 – Each term must contain all variables

$$y + x = y + x + zz' = (x + y + z)(x + y + z')$$

$$x' + z = x' + z + yy' = (x' + y + z)(x' + y' + z)$$

$$y + z = y + z + xx' = (x + y + z)(x' + y + z)$$

step 3 – AND all new terms, eliminating duplicates

$$F(x,y,z) = (x + y + z)(x + y + z')(x' + y + z)(x' + y' + z)$$

$$= (M_0 M_1 M_4 M_6)$$

$$= \Pi(0, 1, 4, 6)$$

Conversion between Canonical Forms

The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function. This is because the original function is expressed by those minterms that make the function equal to 1, whereas its complement is a 1 for those minterms that the function is 0.

Example : $F(A,B,C) = \Sigma(0, 2, 4, 6, 7)$
 $F'(A,B,C) = \Sigma(1, 3, 5) = m_1 + m_3 + m_5$

Take the complement of F' by DeMorgan's theorem to obtain F in a different form:

$$F(A,B,C) = (m_1 + m_3 + m_5)' = (m_1' \cdot m_3' \cdot m_5') = M_1M_3M_5 = \Pi(1, 3, 5)$$

- To convert from one canonical form to the other, interchange the symbols Σ and Π , and list those numbers missing from the original form.

Minimization of Boolean expressions:-

After obtaining SOP and POS expressions, the next step is to simplify the Boolean expression.

There are two methods of simplification of Boolean expressions.

1. Algebraic Method

2. Karnaugh Map :

1. Algebraic method: This method makes use of Boolean postulates, rules and theorems to simplify the expression.

Example..1. Simplify $AB'CD + A'BCD' + ABCD + ABCD'$

Solution-- $AB'CD + AB'CD' + ABCD + ABCD'$

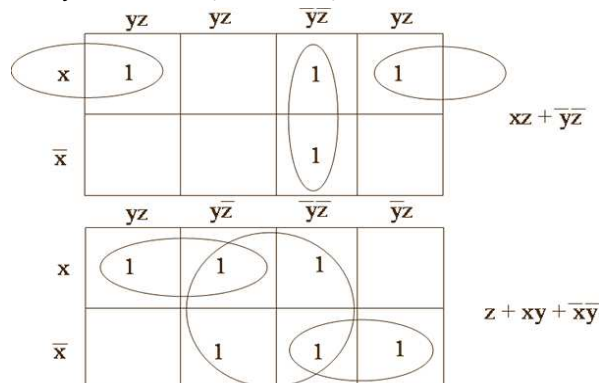
$$\begin{aligned} &= AB'C(D+D') + ABC(D+D') \\ &= AB'C \cdot 1 + ABC \cdot 1 && (D+D'=1) \\ &= AC(B'+B) \\ &= AC \cdot 1 = AC \end{aligned}$$

2. Using Karnaugh Map :

A Karnaugh map is graphical display of the fundamental products in a truth table.

For example:

- Put a 1 in the box for any minterm that appears in the SOP expansion.
- Basic idea is to cover the **largest adjacent** blocks you can whose side length is some power of 2.
- Blocks can "wrap around" the edges.
- For example, the first K-map here represents $xy + x\bar{y} = x(y + \bar{y}) = x$. (since $y+y'=1$)
- The second K-map, similarly, shows $xy + \bar{x}y = (x + \bar{x})y = y$



- Remember, group together adjacent cells of 1s, to form largest possible rectangles of sizes that are powers of 2.
- Notice that you can overlap the blocks if necessary.

Sum Of Products Reduction using K- Map

For reducing the expression first mark Octet, Quad, Pair then single.

- Pair: Two adjacent 1's makes a pair.
- Quad: Four adjacent 1's makes a quad.
- Octet: Eight adjacent 1's makes an Octet.
- Pair removes one variable.
- Quad removes two variables.
- Octet removes three variables.

Reduction of expression: When moving vertically or horizontally in pair or a quad or an octet it can be observed that only one variable gets changed that can be eliminated directly in the expression.

For Example

Q1. Reduce the following Boolean Expression using K-Map:

$$F(A, B, C, D) = \Sigma (0, 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14)$$

	CD			
AB	C'D'	C'D	CD	CD'
A'B'	1		1	1
A'B	1		1	1
AB	1	1		1
AB'	1	1		1

There are 1 octet, 2 quads after eliminating the redundant groups.

Octet ($m_0, m_2, m_4, m_6, m_8, m_{10}, m_{12}, m_{14}$) reduces to D'

Quad (m_2, m_3, m_6, m_7) reduces to $A'C$

Quad (m_8, m_9, m_{12}, m_{13}) reduces to AC'

Hence, $F(A, B, C, D) = D' + A'C + AC'$

Q-2. Reduce the following Boolean Expression using K-Map:

$$F(A, B, C, D) = \Sigma (2, 3, 4, 5, 10, 11, 13, 14, 15)$$

	CD			
AB	C'D'	C'D	CD	CD'
A'B'	0	1	3	2
A'B	4	5	7	6
AB	12	13	15	14
AB'	8	9	11	10

There are 2 quads and 2 pairs after eliminating the redundant groups.

Quad 1 (m₂, m₃, m₁₀, m₁₁) reduces to B'C

Quad 2 (m₁₀, m₁₁, m₁₄, m₁₅) reduces to AC

Pair 1 (m₄, m₅) reduces to A'BC'

Pair 1 (m₁₃, m₁₅) reduces to ABD

Hence, $F(A, B, C, D) = B'C + AC + A'BC' + ABD$

Q4. Reduce the following Boolean Expression using K-Map: $F(A,B,C,D)=\Sigma(0,1,2,4,5,6,8,10)$

Ans:

	C'D'	C'D	CD	CD'
A'B'	1	1		1
A'B	1	1		1
AB				
AB'	1			1

There are 3 quads after eliminating the redundant groups.

Quad 1 (m₀, m₁, m₄, m₅) reduces to A'C'

Quad 2 (m₀, m₂, m₄, m₆) reduces to A'D'

Quad 3 (m₀, m₂, m₈, m₁₀) reduces to B'D'

Hence, $F(A, B, C, D) = A'C' + A'D' + B'D'$

Q3. Reduce the following Boolean Expression using K-Map: $F(A,B,C,D)=\Sigma(0,1,4,5,7,10,13,15)$

Ans:

	C'D'	C'D	CD	CD'
A'B'	1	1		
A'B	1	1	1	
AB		1	1	
AB'				1

There are 2 quads and one block after eliminating the redundant groups.

Quad 1 (m_0, m_1, m_4, m_5) reduces to $A' C'$

Quad 2 (m_0, m_2, m_4, m_6) reduces to $B D$

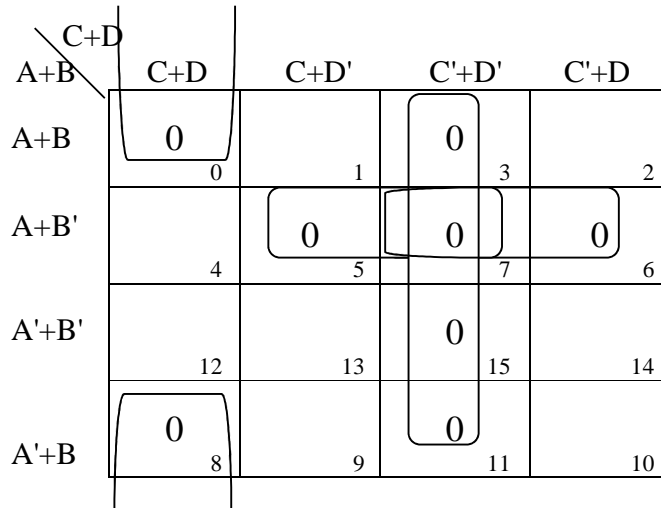
$m_{10} = A B' C D'$

Hence, $F(A, B, C, D) = A' C' + B D + A B' C D'$

Product of Sums Reduction using K-Map

Q2. Reduce the following Boolean Expression using K-Map:

$$F(A, B, C, D) = \pi(0, 3, 5, 6, 7, 8, 11, 15)$$



There are 1 quad and 3 pairs after eliminating the redundant groups.

Quad (M_3, M_7, M_{11}, M_{15}) reduces to $C' + D'$

Pair (M_5, M_7) reduces to $A + B' + D'$

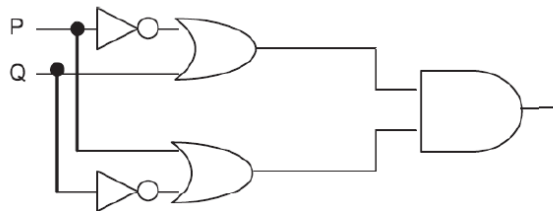
Pair (M_6, M_7) reduces to $A + B' + C'$

Pair (M_0, M_8) reduces to $B + C + D$

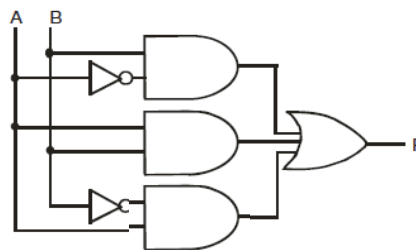
Hence, $F(A, B, C, D) = (C' + D') \cdot (A + B' + D') \cdot (A + B' + C') \cdot (B + C + D)$

2 Marks Questions

1. Write the equivalent Boolean Expression for the following Logic Circuit



2. Write the equivalent Boolean Expression F for the following circuit diagram :



3. Write the equivalent Boolean Expression F for the following circuit diagram :



4. Convert the following Boolean expression into its equivalent Canonical Sum of Product Form((SOP)

$$(X'+Y+Z').(X'+Y+Z).(X'+Y'+Z).(X'+Y'+Z')$$

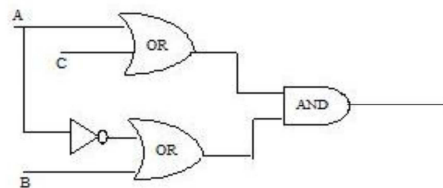
5. Convert the following Boolean expression into its equivalent Canonical Product of Sum form (POS):

$$A.B'.C + A'.B.C + A'.B.C'$$

6. Draw a Logical Circuit Diagram for the following Boolean expression:

$$A.(B+C')$$

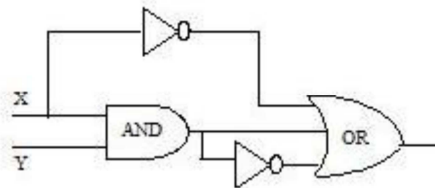
7. Write the equivalent Boolean Expression F for the following circuit diagram:



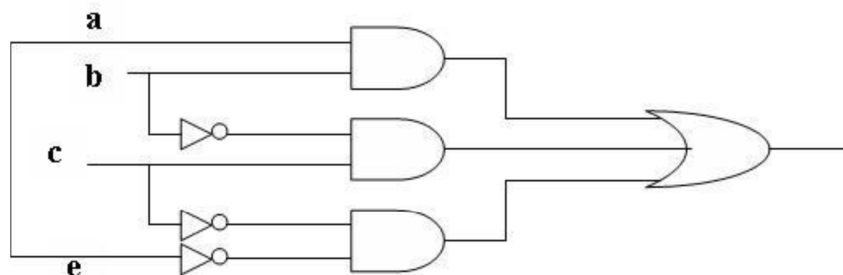
8. Prove that $XY+YZ+YZ'=Y$ algebraically

9. Express the $F(X,Z)=X+X'Z$ into canonical SOP form.

10. Write the equivalent Boolean Expression for the following Logic Circuit.



11. Interpret the following logical circuit as Boolean expression



12. Design $(A+B).(C+D)$ using NAND Gate

13. Prove $x'.y'+y.z = x'yz+x'y'z'+xyz+x'yz$ algebraically.

14. Prove that $(a'+b')(a'+b)(a+b)=a'b'$.

15. A Boolean function F defined on three input variable X,Y,Z is 1 if and only if the number of 1(One) input is odd (e.g. F is 1 if $X=1,Y=0,Z=0$). Draw the truth table for the above function and express it in canonical sum of product form.

3 Marks Questions : Boolean Algebra

1. If $F(a,b,c,d)=\sum(0,2,4,5,7,8,10,12,13,15)$, obtain the simplified form using K-Map.

2. If $F(a,b,c,d) =\sum(0,3,4,5,7,8,9,11,12,13,15)$, obtain the simplified form using KMap

3. Obtain a simplified form for a boolean expression

$$F(U,V,W,Z) = \pi(0,1,3,5,6,7,10,14,15)$$

4. Reduce the following boolean expression using K-Map

$$F(A,B,C,D) = \sum(5,6,7,8,9,12,13,14,15)$$

Answers: 2 Marks Questions : Boolean Algebra

1. $F(P,Q) = (P'+Q).(P+Q')$

2. $A'B+AB+AB'$

3. $X'(Y'+Z)$

4. $F(X, Y, Z) = \pi(4, 5, 6, 7)$

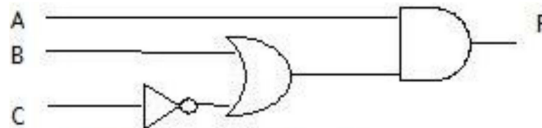
$$= \sum(0, 1, 2, 3)$$

$$= X'. Y'. Z' + X'. Y'. Z + X'. Y. Z' + X'. Y. Z$$

5. $A.B'.C + A'.B.C + A'.B.C' = \pi(0,1,4,6,7)$

OR $= (A+B+C).(A+B+C').(A'+B+C).(A'+B'+C).(A'+B'+C')$

6.



7. $(A+C)(A'+B)$

8. $XY+YZ+YZ'=Y$

L.H.S.

$$XY+YZ+YZ'$$

$$= XY+Y(Z+Z')$$

$$= XY+Y=Y(X+1)$$

$$= Y.1$$

$$= Y=RHS$$

9. $F(X,Z) = X+X'Z = X(Y+Y')+X'(Y+Y')Z$

$$= XY+XY'+X'YZ+X'Y'Z$$

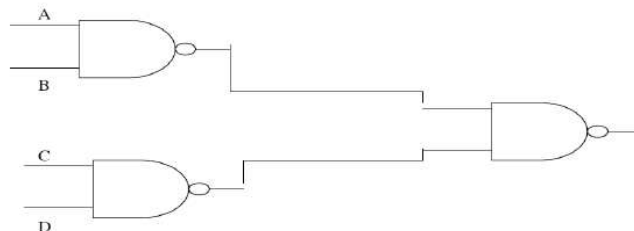
$$= XY(Z+Z')+X'Y'(Z+Z')+X'YZ+X'Y'Z$$

$$= XYZ+XYZ'+X'Y'Z+X'Y'Z'+X'YZ+X'Y'Z$$

10. $XY+(XY)'+X'$

11. $ab+b'c+c'e'$

12.



13. L.H.S. = $x'y + y.z$

$$= x'y.1 + 1.y.z = x'y(z+z') + (x+x')y.z$$

$$= x'yz + x'yz' + xyz + x'yz = RHS$$

14. LHS = $(a'+b')(a'+b)(a+b')$

$$= (a'a' + a'b + a'b' + b'b)(a+b')$$

$$= (a' + a'b + a'b' + 0)(a+b')$$

$$= aa' + a'b + aa'b + a'bb' + a'ab' + a'b'b'$$

$$= 0 + a'b + 0 + 0 + 0 + a'b' = a'b' = RHS$$

15.

XYZ	F
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

Canonical SOP

$$XYZ'+XY'Z+XY'Z'+XYZ$$

Answers: 3 marks Questions Boolean Algebra

1.

$$F(a,b,c,d)=\sum(0,2,4,5,7,8,10,12,13,15)$$

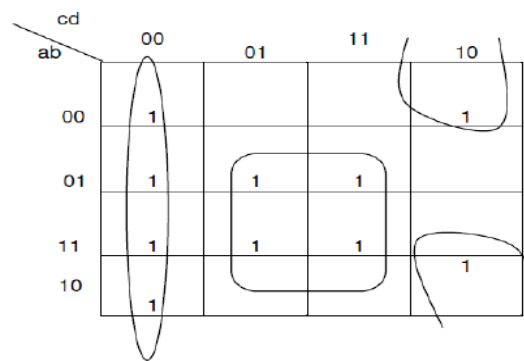
$$F(a,b,c,d)=B1+B2+B3$$

$$B1=m0+m4+m12+m8=c'd'$$

$$B2=m5+m7+m13+m15=bd$$

$$B3=m0+m2+m8+m10=b'd'$$

$$F(a,b,c,d)=c'd'+bd+b'd'$$



2. $F(a,b,c,d) = \sum(0,3,4,5,7,8,9,11,12,13,15)$

$$F(a,b,c,d)=B1+B2+B3+B4$$

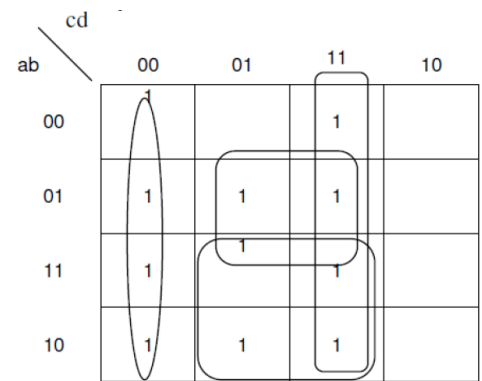
$$B1=m0+m4+m12+m8=c'd'$$

$$B2=m5+m7+m13+m15=bd$$

$$B3=m13+m15+m9+m11=ad$$

$$B4=m3+m7+m15+m11=cd$$

$$F(a,b,c,d)=c'd'+bd+ad+cd$$



3. $F(U,V,W,Z) = \pi(0,1,3,5,6,7,10,14,15)$

$$F(U,V,W,Z)=B1.B2.B3.B4$$

$$B1=M0.M1=(U+V+W)$$

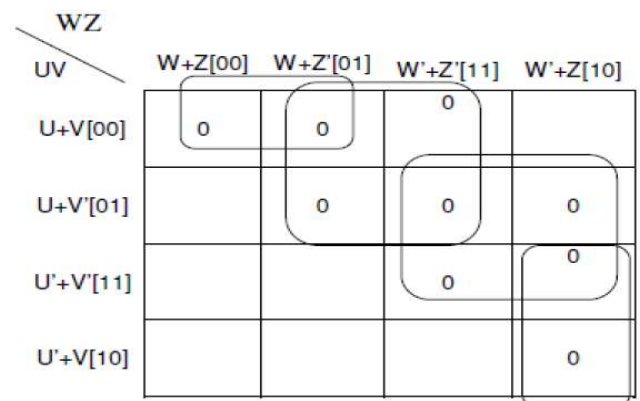
$$B2=M1.M3.M5.M7=(U+Z')$$

$$B3=M7.M6.M15.M14=(V'+W')$$

$$B4=M14.M10=(U'+W'+Z)$$

$$F(U,V,W,Z)=$$

$$(U+V+W).(U+Z').(V'+W').(U'+W'+Z)$$



4. $F(A,B,C,D) = \sum(5,6,7,8,9,12,13,14,15)$

$F(A,B,C,D) = B1 + B2 + B3$

$B1 = M0 + M2 + M8 + M10 = B'D'$

$B2 = M0 + M1 + M4 + M5 = A'C'$

$B3 = M8 + M9 + M11 + M10 = AB'$

$F(A,B,C,D) = B'D' + A'C' + AB'$

	A'.B'	A'.B	A.B	A.B'
C'.D'	1	1		1
	0	4	12	8
C'.D	1	1		1
	1	5	13	9
C.D				1
	3	7	15	11
C.D'	1			1
	2	6	14	10

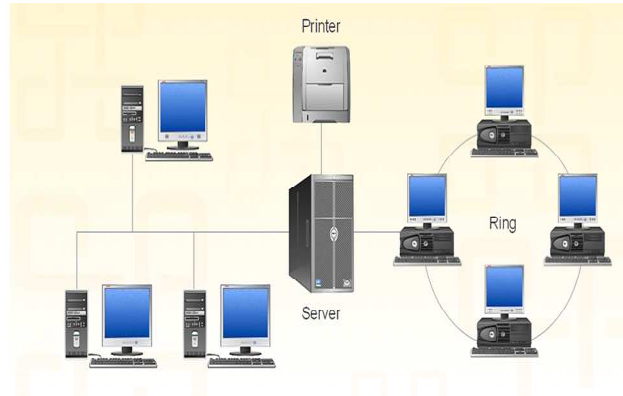
$F(A,B,C,D) = A'.C' + A.B' + B'.D'$

UNIT 5 : COMMUNICATION AND NETWORK CONCEPTS

Points to remember

Network

- The collection of interconnected computers is called a computer network.
- Two computers are said to be interconnected if they are capable of sharing and exchanging information.



Needs

- Resource Sharing
- Reliability
- Cost Factor
- Communication Medium

Resource Sharing means to make all programs, data and peripherals available to anyone on the network irrespective of the physical location of the resources and the user.

Reliability means to keep the copy of a file on two or more different machines, so if one of them is unavailable (due to some hardware crash or any other) then its other copy can be used.

Cost factor means it greatly reduces the cost since the resources can be shared

Communication Medium means one can send messages and whatever the changes at one end are done can be immediately noticed at another.

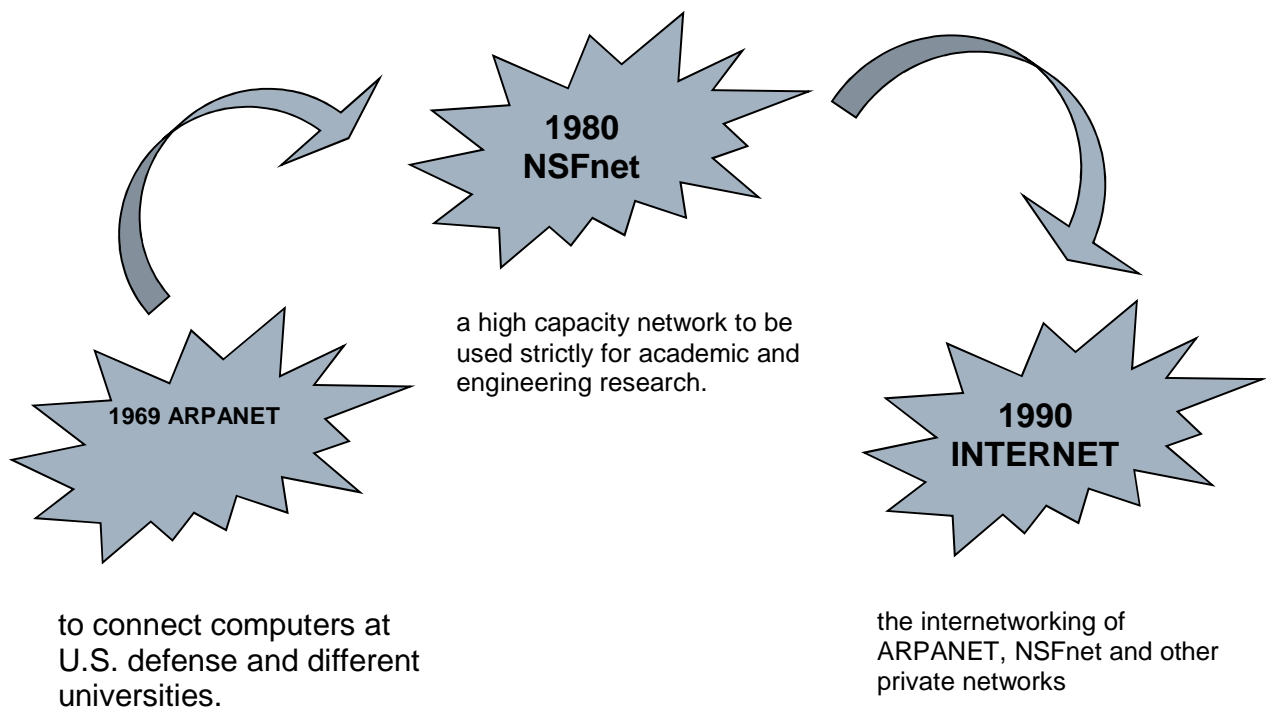
Evolution Of Networking

1969 - First network came into existence

ARPANET (ADVANCED RESEARCH PROJECT AGENCY NETWORK)

MID 80'S - **NSFNET** (NATIONAL SCIENCE FOUNDATION NETWORK)

Diagrammatic presentation



□ **Internet** is the network of networks.

SWITCHING TECHNIQUES

Switching techniques are used for transmitting data across networks.

Different types are :

- Circuit Switching
- Message Switching
- Packet Switching

Circuit Switching

- *Circuit switching* is the transmission technology that has been used since the first communication networks in the nineteenth century.
- First the complete physical connection between two computers is established and then the data are transmitted from the source computer to the destination.
- When a call is placed the switching equipment within the system seeks out a physical copper path all the way from the sender to the receiver.
- It is must to setup an end-to-end connection between computers before any data can be sent.
- The circuit is *terminated* when the connection is closed.
- In circuit switching, resources remain allocated during the full length of a communication, after a circuit is established and until the circuit is terminated and the allocated resources are freed.

Message Switching

- In this the source computer sends data or the message to the switching circuit which stores the data in its buffer.
- Then using any free link to the switching circuit the data is sent to the switching circuit.
- Entire message is sent to the destination. It reaches through different intermediate nodes following the “store and forward” approach.
- No dedicated connection is required.

Packet Switching

- Conceived in the 1960's, *packet switching* is a more recent technology than circuit switching.
- Packet switching introduces the idea of cutting data i.e. at the source entire message is broken in smaller pieces called packets which are transmitted over a network without any resource being allocated.
- Then each packet is transmitted and each packet may follow any rout available and at destination packets may reach in random order.
- If no data is available at the sender at some point during a communication, then no packet is transmitted over the network and no resources are wasted.
- At the destination when all packets are received they are merged to form the original message.
- In packet switching all the packets of fixed size are stored in main memory.

DATA COMMUNICATION TERMINOLOGIES

Data channel	<ul style="list-style-type: none">• The information / data carry from one end to another in the network by channel.
Baud & bits per second (bps)	<ul style="list-style-type: none">• It's used to measurement for the information carry of a communication channel.• Measurement Units :-• bit• 1 Byte= 8 bits• 1 KBPS (Kilo Byte Per Second)= 1024 Bytes ,• 1 Kbps (kilobits Per Second) = 1024 bits,• 1 Mbps (Mega bits Per Second)=1024 Kbps
Bandwidth	<ul style="list-style-type: none">• It is amount of information transmitted or receives per unit time.• It is measuring in Kbps/Mbps etc unit.

Transmission Media

- data is transmitted over copper wires, fiber optic cable, radio and microwaves. the term 'media' is used to generically refer to the physical connectors, wires or devices used to plug things together.
- **Basic communications media types**
- copper
 - unshielded twisted pair (utp)
 - shielded twisted pair (stp)
 - coaxial cable (thinnet, thicknet)
- fiber optic

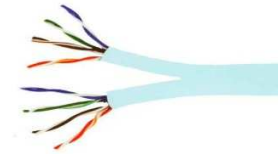
- single-mode
- multi-mode
- infrared
- radio & microwave

Twisted Pair Cable

- These cables consist of two insulated copper wires twisted around each other in a double helix.
- Twisting of wires reduces crosstalk which is bleeding of a signal from one wire to another.

Types:

- Unshielded Twisted Pair (UTP)
- Shielded Twisted Pair (STP)



STP offers greater protection from interference and crosstalk due to shielding. But it is heavier and costlier than UTP.

USE:

1. In local telephone communication
2. For digital data transmission over short distances upto 1 km

Advantages:

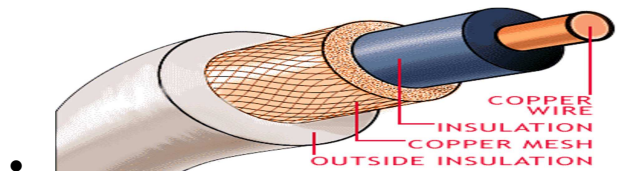
- Easy to install and maintain
- Simple
- Inexpensive
- Low weight
- Suitable for small (Local) Networks

Disadvantages:

- Not suitable for long distance due to high attenuation.
- Low bandwidth support.
- Low Speed

Coaxial cable

- Coaxial cable consists of a solid copper wire core surrounded by a plastic cladding shielded in a wire mesh.
- Shield prevents the noise by redirecting it to ground.



Types:

Coaxial cable comes in two sizes which are called *thinnet* and *thicknet*.

- Thicknet : segment length upto 500 m
- Thinnet : segment length upto 185 m

USE:

In TV channel communication

Advantages:

- Better than twisted wire cable.
- Popular for TV networks.
- Offers higher bandwidth & Speed

Disadvantages:

- Expensive than twisted wires.
- Not compatible with twisted wire cable.

Optical Fibres

- Thin strands of glass or glass like material designed to carry light from one source to another.
- Source converts (Modulates) the data signal into light using LED (Light Emitting Diodes) or LASER diodes and send it over the Optical fiber.

It consists of three parts:

1. The core: glass or plastic through which the light travels.
2. The cladding : covers the core and reflects light back to the core
3. Protective coating : protects the fiber

Advantages

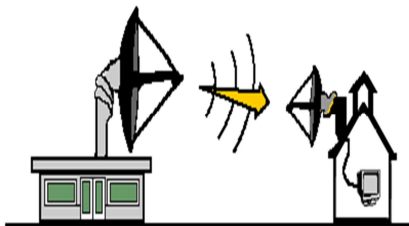
- Not affected by any kind of noise.
- High transmission capacity
- Speed of Light
- Suitable for broadband communication

Disadvantages

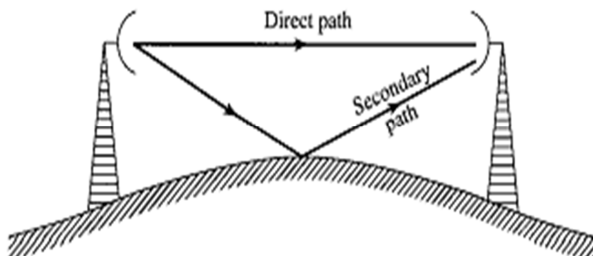
- Installation requires care.
- Connecting two Optical fibers is difficult.
- Optical fibers are more difficult to solder
- Most expensive

Microwaves

Microwaves are transmitted from the transmitters placed at very high towers to the receivers at a long distance.



Microwaves are transmitted in line of sight fashion, and also propagated through the surfaces.



Advantages

- Maintenance easy than cables.

- Suitable when cable can not be used.

-

Disadvantages

- Repeaters are required for long distance communication.
- Less Bandwidth available.

Satellite

Geostationary satellites are placed around 36000 KM away from the earth's surface. In satellite communication transmitting station transmits the signals to the satellite. (It is called up-linking). After receiving the signals (microwaves) it amplifies them and transmit back to earth in whole visibility area. Receiving stations at different places can receive these signals. (It is called down-linking).



Advantage

- Area coverage is too large

Disadvantage

- High investment

Network devices

Modem

- A modem is a computer peripheral that allows you to connect and communicate with other computers via telephone lines.
- Modem means Modulation/ Demodulation.
- Modulation: A modem changes the digital data from your computer into analog data, a format that can be carried by telephone lines.
- Demodulation: The modem receiving the call then changes the analog signal back into digital data that the computer can digest.
- The shift of digital data into analog data and back again, allows two computers to speak with one another.

External Modem



Internal Modem



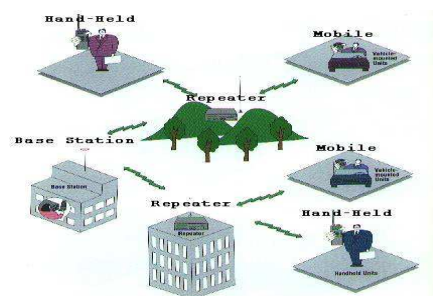
RJ-45 Connector

RJ-45 is short for Registered Jack-45. It is an eight wire connector which is commonly used to connect computers on the local area networks i.e., LAN.

Network Interface Cards (Ethernet Card)

- A network card, network adapter or NIC (network interface card) is a piece of computer hardware designed to allow computers to communicate over a **computer network**. It provides physical access to a networking medium and often provides a low-level addressing system through the use of MAC addresses. It allows users to connect to each other either by using cables or wirelessly.

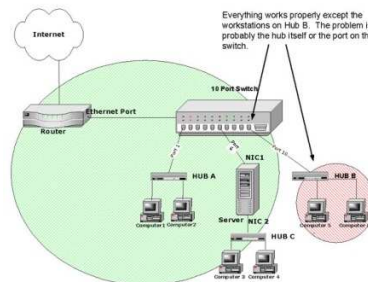
Repeaters



A hub contains multiple ports. one port, it is copied to all the packets are copied, the destination not change to a broadcast address. way, it simply copies the data to to the hub.

A repeater is an electronic device that receives a signal and retransmits it at a higher level or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances without degradation. In most twisted pair Ethernet configurations, repeaters are required for cable runs longer than 100 meters.

Hubs



When a packet arrives at ports of the hub. When the address in the frame does It does this in a rudimentary all of the Nodes connected



BridgesA network bridge connects multiple network segments at the data link layer (layer 2) of the OSI model. Bridges do not promiscuously copy traffic to all ports, as hubs do, but learn which MAC addresses are reachable through specific ports. Once the bridge associates a port and an address, it will send traffic for that address only to that port. Bridges do send broadcasts to all ports except the one on which the broadcast was received.

- Bridges learn the association of ports and addresses by examining the source address of frames that it sees on various ports. Once a frame arrives through a port, its source address is stored and the bridge assumes that MAC address is associated with that port. The first time that a previously unknown destination address is seen, the bridge will forward the frame to all ports other than the one on which the frame arrived.

- Bridges come in three basic types:
- Local bridges: Directly connect local area networks (LANs)
- Wireless bridges: Can be used to join LANs or connect remote stations to LANs.

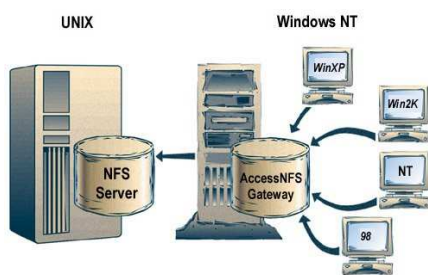
Switches

Switch is a device that performs switching. Specifically, it forwards and filters OSI layer 2 datagrams (chunk of data communication) between ports (connected cables) based on the Mac-Addresses in the packets. This is distinct from a hub in that it only forwards the datagrams to the ports involved in the communications rather than all ports connected. Strictly speaking, a switch is not capable of routing traffic based on IP address (layer 3) which is necessary for communicating between network segments or within a large or complex LAN. Some switches are capable of routing based on IP addresses but are still called switches as a marketing term. A switch normally has numerous ports with the intention that most or all of the network be connected directly to a switch, or another switch that is in turn connected to a switch.

Routers

- Routers are networking devices that forward data packets between networks using headers and forwarding tables to determine the best path to forward the packets. Routers work at the network layer of the TCP/IP model or layer 3 of the OSI model. Routers also provide interconnectivity between like and unlike media (RFC 1812).
- A router is connected to at least two networks, commonly two LANs or WANs or a LAN and its ISP's network.

GATEWAY

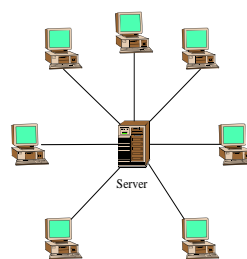


A Gateway is a network device that connects dissimilar networks. It established an intelligent connection between a local area network and external networks with completely different structures.

Network topologies and types

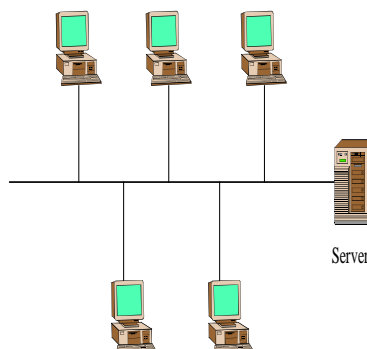
Star Topology

nodes of the network is connected to a central node with a point-to-point link in a 'hub' and the 'spoke' fashion, the central node being attached to the central node being the point-to-point links from the peripheral node) – all data that is transmitted transmitted to this central node, which that then retransmits the data to some or network, although the central node may connection point (such as a 'punch-device to repeat the signals.



The type of network topology in which each of the the 'hub' and the nodes that are 'spokes' (e.g., a collection of nodes that converge at a central between nodes in the network is is usually some type of device all of the other nodes in the also be a simple common down' block) without any active

Bus Topology

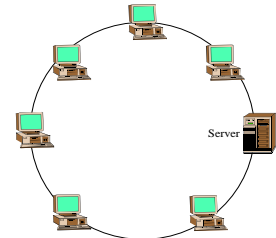


- The type of network topology in which all of the nodes of the network are connected to a common transmission medium which has exactly two endpoints (this is the 'bus', which is also commonly referred to as the backbone, or trunk) – all data that is transmitted between nodes in the network is transmitted over this common transmission medium and is able

to be received by all nodes in the network virtually simultaneously (disregarding propagation delays).

Ring Topology

- The type of network topology in which each of the nodes of the network is connected to two other nodes in the network and with the first and last nodes being connected to each other, forming a ring – all data that is transmitted between nodes in the network travels from one node to the next node in a circular manner and the data generally flows in a single direction only.



Computer Networks

- A communications network is two or more computers connected to share data and resources are “networked.” The simple idea behind computer networking is to allow users to access more information and give them access to devices not directly attached to their “local” system, such as printers or storage devices

Local Area Network (LAN)

A network covering a small geographic area, like a home, office, or building. Current LANs are most likely to be based on Ethernet technology. For example, a library will have a wired or a communications network is two or more computers connected to share data and resources are “networked.” The simple idea behind computer networking is to allow users to access more information and give them access to devices not directly attached to their “local” system, such as **printers or storage devices**

Metropolitan Area Network (MAN)

- A Metropolitan Area Network is a network that connects two or more Local Area Networks or Campus Area Networks together but does not extend beyond the boundaries of the immediate town, city, or metropolitan area. Multiple routers, switches & hubs are connected to create a MAN.



Wide Area Network (WAN)

- WAN is a data communications network that covers a relatively broad geographic area (i.e. one city to another and one country to another country) and that often uses transmission facilities provided by common carriers, such as telephone companies. WAN technologies generally function at the lower three layers of the OSI reference model: the physical layer, the data link layer, and the network layer.

Network protocol

Protocols

- A protocol means the rules that are applicable for a network.

- It defines the standardized format for data packets, techniques for detecting and correcting errors and so on.
- A protocol is a formal description of message formats and the rules that two or more machines must follow to exchange those messages.
- E.g. using library books.

Types of protocols are:

1. HTTP
 2. FTP
 3. TCP/IP
 4. SLIP/PPP
- **Hypertext Transfer Protocol (HTTP)** is a communications protocol for the transfer of information on the intranet and the World Wide Web. Its original purpose was to provide a way to publish and retrieve hypertext pages over the Internet.
 - HTTP is a request/response standard between a client and a server. A client is the end-user; the server is the web site.
 - **FTP (File Transfer Protocol)** is the simplest and most secure way to exchange files over the Internet. The objectives of FTP are:
 - To promote sharing of files (computer programs and/or data).
 - To encourage indirect or implicit use of remote computers.
 - To shield a user from variations in file storage systems among different hosts.
 - To transfer data reliably, and efficiently.
 - **TCP/IP (Transmission Control Protocol / Internet Protocol)**

TCP - is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.

IP - is responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world.

TCP and IP were developed by a Department of Defense (DOD) research project to connect a number different networks designed by different vendors into a network of networks. Several computers in a small department can use TCP/IP (along with other protocols) on a single LAN. The IP component provides routing from the department to the enterprise network, then to regional networks, and finally to the global Internet.

- **SLIP/PPP (Serial Line Internet Protocol / Point to Point Protocol)**

SLIP/PPP provides the ability to transport TCP/IP traffic over serial line between two computers. The home user's computer has a communications link to the internet. The home user's computer has the networking software that can speak TCP/IP with other computers on the Internet. The home user's computer has an identifying address (IP address) at which it can be contacted by other computers on Internet. E.g. dial up connection.

SMTP :- SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with one of two other protocols, POP3 or IMAP, that let the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving e-mail. On Unix-based systems, sendmail is the most widely-used SMTP server for e-mail. A commercial package, Sendmail, includes a POP3 server. Microsoft Exchange includes an SMTP server and can also be set up to include POP3 support.

POP3:- POP3 (Post Office Protocol 3) is the standard way which has been around for decades. It is very similar regular mail. Messages are delivered to our computer, put in our mailbox, and are then our responsibility. Advantage of POP3 :

- 1) Email is available when we are offline
- 2) Email is not stored on the server, so our disk usage on the server is less
- 3) Just about any email client (software) supports POP3

IMAP:- IMAP (Interactive Mail Access Protocol). It is not like the mailbox on your house. With IMAP mail is delivered to the server, and we connect to the server to see our mail. The mail is not stored on our machine. When a message is marked as read, it is marked as read on the server, not on our computer.

Advantages of IMAP:-

- 1) Email is available from any machine we happen to use
- 2) Email is stored on the server, so our email cannot be deleted/destroyed if our computer should happen to crash, be stolen, or destroyed
- 3) We can access IMAP mail via the web, without even needing a mail client installed. This means We can check our mail from someone else's machine or even a public terminal and not have to worry about the security of your passwords.

Telnet-It is an older internet utility that lets us log on to remote computer system. It also facilitates for terminal emulation purpose. Terminal emulation means using a pc like a mainframe computer through networking.

- (i) Run telnet client- Type telnet in run dialog box.
- (ii) Connect to telnet site -specify the host name, port and terminal type.
- (iii) Start browsing- surf the shown site with provided instruction.
- (iv) Finally disconnect-press Alt+F4.

Wireless/Mobile Computing

Wireless communication is simply data communication without the use of landlines. Mobile computing means that the computing device is not continuously connected to the base or central network.

1. **GSM(Global System for Mobile communication)**: it is leading digital cellular system. In covered areas, cell phone users can buy one phone that will work any where the standard is supported. It uses narrowband TDMA, which allows eight simultaneous calls on the same radio frequency.
2. **CDMA(Code Division Multiple Access)**: it is a digital cellular technology that uses spread-spectrum techniques. CDMA does not assign a specific frequency to each user. Instead ,every channel uses the full available spectrum.
3. **WLL(Wireless in Local Loop)** : WLL is a system that connects subscribers to the public switched telephone network using radio signals as a substitute for other connecting media.

4. **Email(Electronic Mail):** Email is sending and receiving messages by computer.
5. **Chat:** Online textual talk in real time , is called Chatting.
6. **Video Conferencing:** a two way videophone conversation among multiple participants is called video conferencing.
7. **SMS(Short Message Service):** SMS is the transmission of short text messages to and from a mobile phone, fax machine and or IP address.
8. **3G and EDGE:** 3G is a specification for the third generation of mobile communication of mobile communication technology. 3G promises increased bandwidth, up to 384 Kbps when a device is stationary.
9. **EDGE(Enhanced Data rates for Global Evolution)** is a radio based high speed mobile data standard.

10.Wi-Fi---Wi-Fi is short for *Wireless Fidelity* .**Wi-fi** is a wireless technology that uses radio frequency to transmit data through the air. An **Access Point** is a separate wireless unit, which has the ability to extend from the router to maximize wireless reception. Up to 16 users can connect to one access point .It allows several wireless clients to connect to a single device.**Wi-fi** hot spot is defined as any location in which wireless technology both exists and is available for use to consumers.

Wi-fi was intended to be used for wireless devices and LANS, but is now often used for internet access.

11.**IPR:-** The term "Intellectual Property Rights" refers to the legal rights granted with the aim to protect the creations of the intellect. These rights include Industrial Property Rights (e.g. patents, industrial designs and trademarks) and Copyright (right of the author or creator) and Related Rights (rights of the performers, producers and broadcasting organizations

12.**Cloud Computing :-** The practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer.

Advantages:-1) Cost efficient 2) Backup & Restore 3) Almost unlimited storage 4) Automatic Software Integration 5) Easy access of Information

Disadvantages:-1) Technical Issue 2)Security in the cloud 3) Prone to attack

NETWORK SECURITY CONCEPTS

Protection methods

1 Authorization - Authorization confirms the service requestor's credentials. It determines if the service requestor is entitled to perform that operation.

2 Authentication - Each entity involved in using a web service the requestor, the provider and the broker(if there is one) - is what it actually claims to be.

3 Encryption – conversion of the form of data from one form to another form.

4 Biometric System - involves unique aspect of a person's body such as Finger-prints, retinal patterns etc to establish his/her Identity.

5 Firewall - A system designed to prevent unauthorized access to or from a private network is called firewall. it can be implemented in both hardware and software or combination or both.

There are several types of firewall techniques-

* **Packet filter-** accepts or rejects of packets based on user defined rules.

* **Application gateway-** security mechanism to specific application like FTP and Telnet servers.

* **Circuit level gateway** - applies security mechanism when a connection is established.

* **Proxy Server** - Intercepts all messages entering and leaving the network.

Cookies - Cookies are messages that a web server transmits to a web browser so that the web server can keep track of the user's activity on a specific web site. Cookies have few parameters name, value, expiration date

Hackers and crackers -

Hackers are more interested in gaining knowledge about computer systems and possibly using this knowledge for playful pranks.

Crackers are the malicious programmers who break into secure systems.

Cyber Law -

It is a generic term, which refers to all the legal and regulatory aspects of internet and the World Wide Web.

WEB SERVERS

WWW (WORLD WIDE WEB)

It is a small part of Internet. It is a kind of Application of internet. It is a set of protocols that allows us to access any document on the Net through a naming system based on URLs. Internet was mainly used for obtaining textual information. But post-WWW the internet popularity grew tremendously because of graphic intensive nature of www.

Attributes of WWW

- (i) **User friendly**- www resources can be easily used with the help of browser.
- (ii) **Multimedia documents**-A web page may have graphic, audio, video, and animation etc at a time.
- (iii) **Hypertext and hyperlinks**-the dynamic links which can move towards another web page is hyperlink.
- (iv) **Interactive** -www with its pages support and enable interactivity between users and servers.
- (v) **frame**-display of more than one section on single web page.

Web server- It is a WWW server that responds to the requests made by web browsers.
e.g. : Apache, IIS, PWS(Personal web server for Windows 98).

Web browser- It is a WWW client that navigates through the World Wide Web and displays web pages. E.g.: FireFox Navigator, Internet Explorer etc.

Web sites- A location on a net server where different web pages are linked together by dynamic links is called a web site. Each web site has a unique address called URL.

Web page - A document that can be viewed in a web browser and residing on a web site is a web page.

Home page- a web page that is the starting page and acts as an indexed page is home page.

Web portal - that facilitates various type of the functionality as web site. for e.g. www.yahoo.com, www.rediff.com

Domain name- An internet address which is a character based is called a Domain name. Some most common domains are com, edu, gov, mil, net, org, and co. Some domain names are location based also. For e.g. au for Australia, a for Canada, in for India etc.

URL- A URL (uniform resource locator) that specifies the distinct address for each resource on the internet.e.g.http://encycle.msn.com/getinfo/stypes.asp

Web hosting - means hosting web server application on a computer system through which electronic content on the internet is readily available to any web browser client.

HTML -

It stands for Hyper Text Markup Language that facilitates to write web document that can be interpreted by any web browser. It provide certain tags that are interpreted by the browser how to display and act with the text, graphics etc. tags are specified in <>.

For e.g.

<body bgcolor=green> it is opening tag

</body> it is closing tag.

body is the tag with bgcolor attributes.

XML (eXtensible Markup Language)

XML is a markup language for documents containing structured information. Structured information contains both content (words, pictures etc.) and some indication of what role content plays.

DHTML- It stands for Dynamic Hyper Text Markup Language. DHTML refers to Web content that changes each time it is viewed. For example, the same URL could result in a different page depending on any number of parameters, such as:

*geographic location

*time of the day

*previous pages viewed by the user

*profile of the reader

WEB SCRIPTING – The process of creating and embedding scripts in a web page is known as web-scripting.

SCRIPT: A Script is a list of commands embedded in a web page. Scripts are interpreted and executed by a certain program or scripting –engine.

Types of Scripts:

1. Client Side Script: Client side scripting enables interaction within a web page.

Some popular client-side scripting languages are VBScript, JavaScript, PHP(Hyper Text Preprocessor).

2. Server-Side Scripts: Server-side scripting enables the completion or carrying out a task at the server-end and then sending the result to the client –end.

Some popula server-side Scripting Languages are PHP, Perl, ASP(Active Server Pages), JSP(Java Server Pages) etc.

OPEN SOURCE TERMINOLOGIES

TERMINOLOGY & DEFINITIONS:

- **Free Software:** The S/W's is freely accessible and can be freely used changed improved copied and distributed by all and payments are needed to make for free S/W.
- **Open Source Software:** S/w whose source code is available to the customer and it can be modified and redistributed without any limitation .OSS may come free of cost but nominal charges has to pay nominal charges (Support of S/W and development of S/W).
- **FLOSS (Free Libre and Open Source Software) :** S/w which is free as well as open source S/W. (Free S/W + Open Source S/W).

- **GNU (GNU's Not Unix) :** GNU project emphasize on the freedom and its objective is to create a system compatible to UNIX but not identical with it.
- **FSF (Free Software Foundation) :** FSF is a non –profit organization created for the purpose of the free s/w movement. Organization funded many s/w developers to write free software.
- **OSI (Open Source Initiative) :** Open source software organization dedicated to cause of promoting open source software it specified the criteria of OSS and its source code is not freely available.
- **W3C(World Wide Web Consortium) :** W3C is responsible for producing the software standards for World Wide Web.
- **Proprietary Software:** Proprietary Software is the s/w that is neither open nor freely available, normally the source code of the Proprietary Software is not available but further distribution and modification is possible by special permission by the supplier.
- **Freeware:** Freeware are the software freely available , which permit redistribution but not modification (and their source code is not available). Freeware is distributed in *Binary Form* (ready to run) without any licensing fees.
- **Shareware:** Software for which license fee is payable after some time limit, its source code is not available and modification to the software are not allowed.
- **Localization:** localization refers to the adaptation of language, content and design to reflect local cultural sensitivities .e.g. Software Localization: where messages that a program presents to the user need to be translated into various languages.
- **Internationalization:** Opposite of localization.

OPEN SOURCE / FREE SOFTWARE

- **Linux :** Linux is a famous computer operating system . popular Linux server set of program –LAMP(Linux, Apache, MySQL, PHP)
- **Mozilla :** Mozilla is a free internet software that includes
 - a web browser
 - an email client
 - an HTML editor
 - IRC client
- **Apache server:** Apache web server is an open source web server available for many platforms such as BSD, Linux, and Microsoft Windows etc.
 - Apache Web server is maintained by open community of developers of Apache software foundation.
- **MYSQL :** MYSQL is one of the most popular open source database system. Features of MYSQL :
 - Multithreading
 - Multi –User
 - SQL Relational Database Server
 - Works in many different platform
- **PostgreSQL :** Postgres SQL is a free software object relational database server . PostgreSQL can be downloaded from www.postgresql.org.
- **Pango :** Pango project is to provide an open source framework for the layout and rendering of internationalized text into GTK + GNOME environment.Pango using

Unicode for all of its encoding ,and will eventually support output in all the worlds major languages.

- **OpenOffice** : OpenOffice is an office applications suite. It is intended to compatible and directly complete with Microsoft office.

OOo Version 1.1 includes:

- Writer (word processor)
- Calc(spreadsheet)
- Draw(graphics program)
- etc
- **Tomcat** : Tomcat functions as a servlet container. Tomcat implements the servlet and the JavaServer Pages .Tomcat comes with the jasper compiler that complies JSPs into servlets.
- **PHP(Hypertext Preprocessor)** : PHP is a widely used open source programming language for server side application and developing web content.
- **Python: Python** is an interactive programming language originally as scripting language for Amoeba OS capable of making system calls.

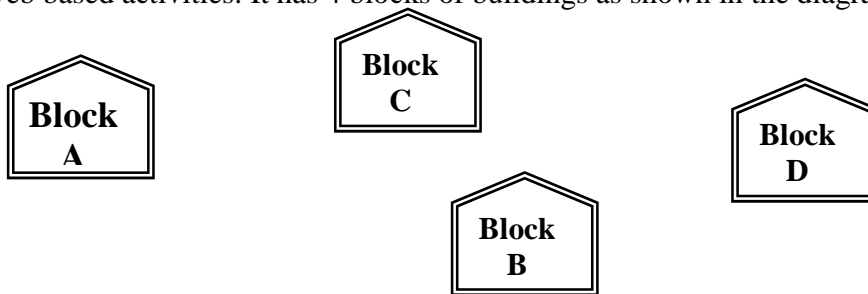
Web 2.0 services : Web 2.0 is the term given to describe a second generation of the World Wide Web that is focused on the ability for people to collaborate and share information online. Web 2.0 basically refers to the transition from static HTML Web pages to a more dynamic Web that is more organized and is based on servicing Web applications to users.

Other improved functionality of Web 2.0 includes open communication with an emphasis on Web-based communities of users, and more open sharing of information. Over time Web 2.0 has been used more as a marketing term than a computer-science-based term. Blogs, wikis, and Web services are all seen as components of Web 2.0.

Web 2.0 was previously used as a synonym for Semantic Web, but while the two are similar, they do not share precisely the same meaning.

Long Answer Questions

Q1. Knowledge Supplement Organisation has set up its new centre at Mangalore for its office and web based activities. It has 4 blocks of buildings as shown in the diagram below:



Centre to centre distances between various blocks

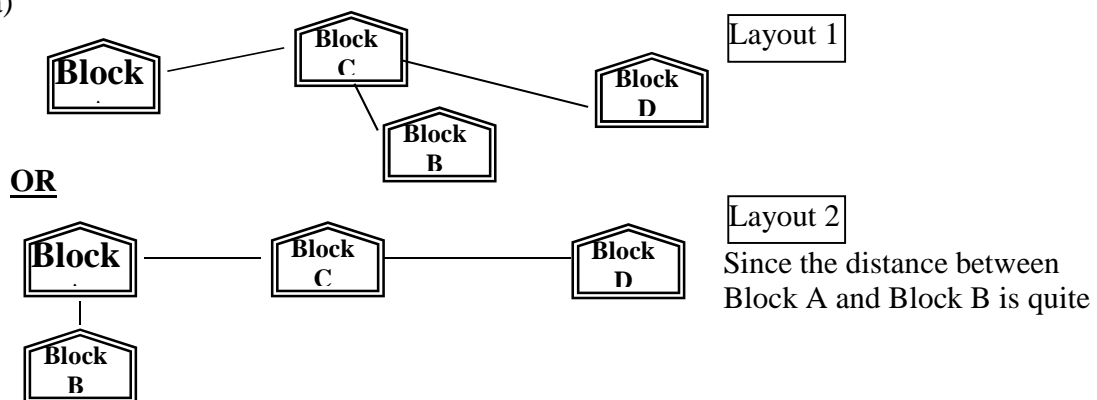
Block A to Block B	50 m
Block B to Block C	150 m
Block C to Block D	25 m
Block A to Block D	170 m
Block B to Block D	125 m
Block A to Block C	90 m

Number of Computers

Block A	25
Block B	50
Block C	125
Block D	10

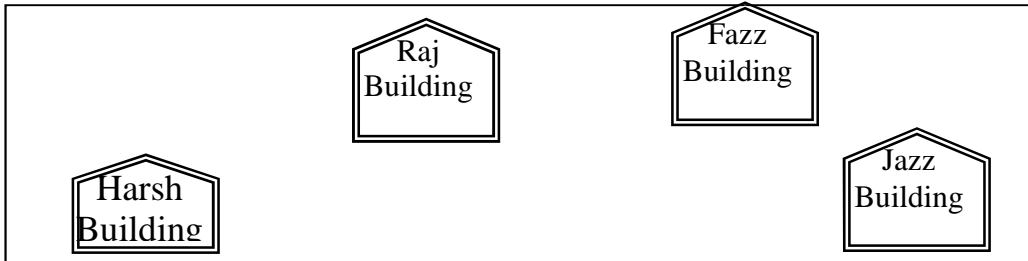
- (a) Suggest a cable layout of connections between the blocks.
- (b) Suggest the most suitable place (i.e. block) to house the server of this organisation with a suitable reason.
- (c) Suggest the placement of the following devices with justification
 - (i) Repeater
 - (ii) Hub/Switch
- (d) The organization is planning to link its front office situated in the city in a hilly region where cable connection is not feasible, suggest an economic way to connect it with reasonably high speed?

Ans : (a)



- (b) The most suitable place / block to house the server of this organisation would be Block C, as this block contains the maximum number of computers, thus decreasing the cabling cost for most of the computers as well as increasing the efficiency of the maximum computers in the network.
- (c) (i) For Layout 1, since the cabling distance between Blocks A and C, and that between B and C are quite large, so a repeater each, would ideally be needed along their path to avoid loss of signals during the course of data flow in these routes.
For layout 2, since the distance between Blocks A and C is large so a repeater would ideally be placed in between this path
- (ii) A hub/switch each would be needed in all the blocks, to interconnect the group of cables from the different computers in each block.
- (a) The most economic way to connect it with a reasonable high speed would be to use radio wave transmission, as they are easy to install, can travel long distances, and penetrate buildings easily, so they are widely used for communication, both indoors and outdoors. Radio waves also have the advantage of being omni directional, which is they can travel in all the directions from the source, so that the transmitter and receiver do not have to be carefully aligned physically.

Q2. Ravya Industries has set up its new center at Kaka Nagar for its office and web based activities. The company compound has 4 buildings as shown in the diagram below:



Centre to centre distances between various buildings

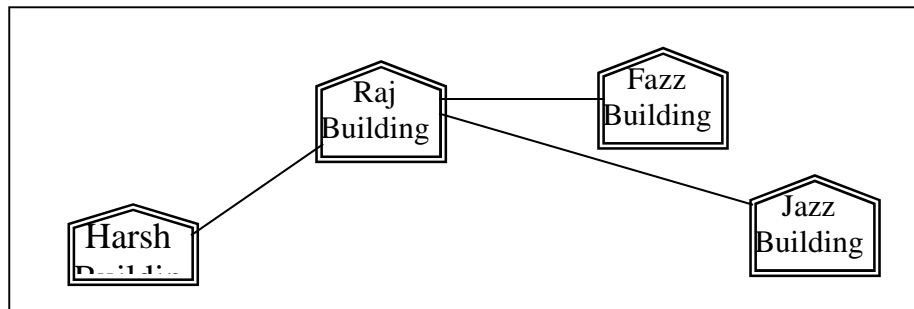
Harsh Building to Raj Building	50 m
Raz Building to Fazz Building	60 m
Fazz Building to Jazz Building	25 m
Jazz Building to Harsh Building	170 m
Harsh Building to Fazz Building	125 m
Raj Building to Jazz Building	90 m

Number of Computers

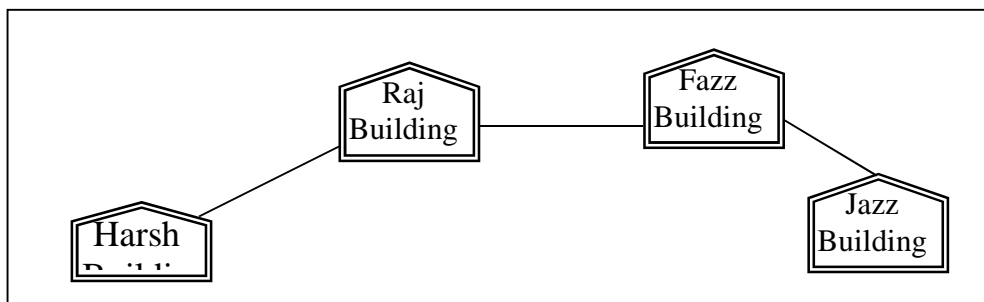
Harsh Building	15
Raj Building	150
Fazz Building	15
Jazz Building	25

- Suggest a cable layout of connections between the buildings.
- Suggest the most suitable place (i.e. building) to house the server of this organisation with a suitable reason.
- Suggest the placement of the following devices with justification:
 - Internet Connecting Device/Modem
 - Switch
- The organisation is planning to link its sale counter situated in various parts of the same city, which type of network out of LAN, MAN or WAN will be formed? Justify your answer.

Ans : (a) Layout 1:



Layout 2: Since the distance between Fazz Building and Jazz Building is quite short



- (b) The most suitable place / block to house the server of this organisation would be Raj Building, as this block contains the maximum number of computers, thus decreasing the cabling cost for most of the computers as well as increasing the efficiency of the maximum computers in the network.
- (c) (i) Raj Building
(ii) In both the layouts, a hub/switch each would be needed in all the buildings, to interconnect the group of cables from the different computers in each block
- (d) The type of network that shall be formed to link the sale counters situated in various parts of the same city would be a MAN, because MAN (Metropolitan Area Networks) are the networks that link computer facilities within a city.

Q3. Nootan Vidya Mandir in OOTY is setting up the network between its different wings. There are 4 wings named as SENIOR(S), MIDDLE(M), JUNIOR(J) and OFFICE(O).

Distance between the various wings are given below:

Wing O to Wing S	100m
Wing O to Wing M	200m
Wing O to Wing J	400m
Wing S to Wing M	300m
Wing S to Wing J	100m
Wing J to Wing M	450m

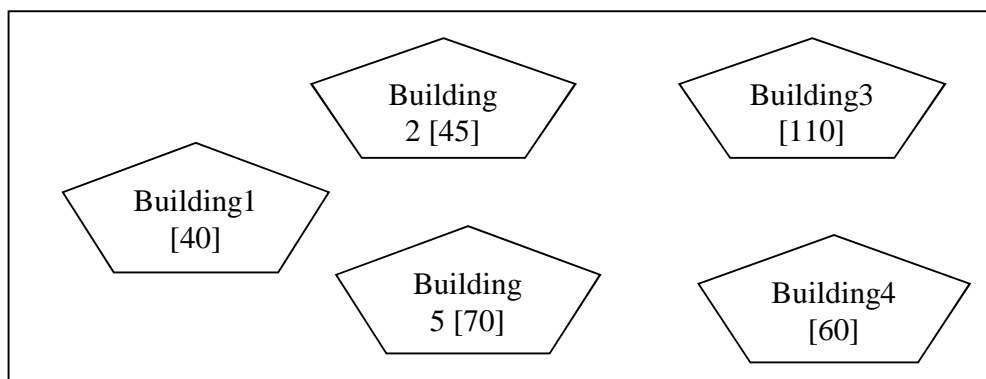
Number of Computers

Wing O	10
Wing S	200
Wing M	100
Wing J	50

- (i) Suggest a suitable Topology for networking the computer of all wings.
- (ii) Name the wing where the server to be installed. Justify your answer.
- (iii) Suggest the placement of Hub/Switch in the network.
- (iv) Mention an economic technology to provide internet accessibility to all wings.

Ans: (i) Star or Bus or any other valid topology.
(ii) Wing S, because maximum number of computers are located at Wing S.
(iii) Hub/ Switch in all the wings.
(iv) Coaxial cable/Modem/LAN/TCP-IP/Dialup/DSL/Leased Lines or any other valid technology.

Q4. East and West Public Ltd. has decided to network all its offices spread in five buildings.



[] indicates the total no. of computers.

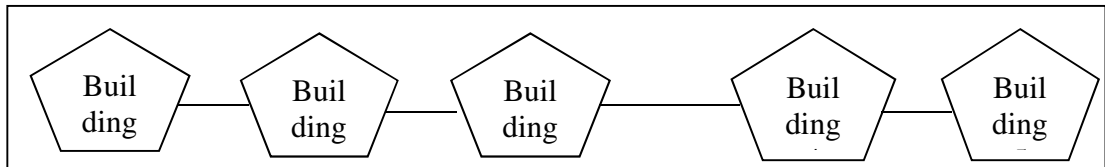
The distances between buildings are given below:

Between 1 & 2	20 Mts
Between 2 & 3	50 Mts
Between 3 & 4	120 Mts
Between 4 & 5	30 Mts

Between 3 & 5	70 Mts
Between 1 & 5	65 Mts
Between 2 & 5	50 Mts
Between 1 & 3	60 Mts

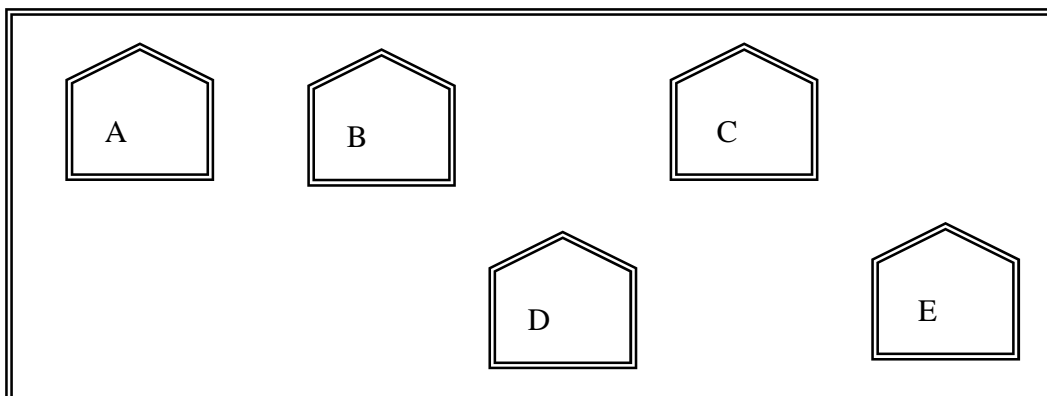
- (i) Suggest cable layout(s) for connecting the buildings.
- (ii) Suggest the most suitable building to install the server of this organization with a suitable reason, with justification.
- (iii) Suggest the placement of the following devices with justification?
 - (a) Hub/Switch
 - (b) Repeater
- (iv) The company also has another office in the same city but at a distant location about 25-30 kms away. How can link be established with this building? (i.e. suggest the transmission medium).

Ans : (i)



- (ii) The server should be installed in building 3 as it contains the maximum number of computers, thus decreasing the cabling cost for most of the computers as well as increasing the efficiency of the maximum computers in the network.
- (iii) (a) Hub/switch each would be needed in all the buildings, to interconnect the group of cables from the different computers in each building.
 (b) Since the cabling distance between Building 3 and Building 4 is quite large, so a repeater would ideally be needed along their path to avoid loss of signals during the course of data flow in these routes.
- (iv) Microwave / Radiowave / Satellite

Q5. Standard Bank has set up its new center in India for its office and web based activities. It has five buildings as shown in the diagram below:

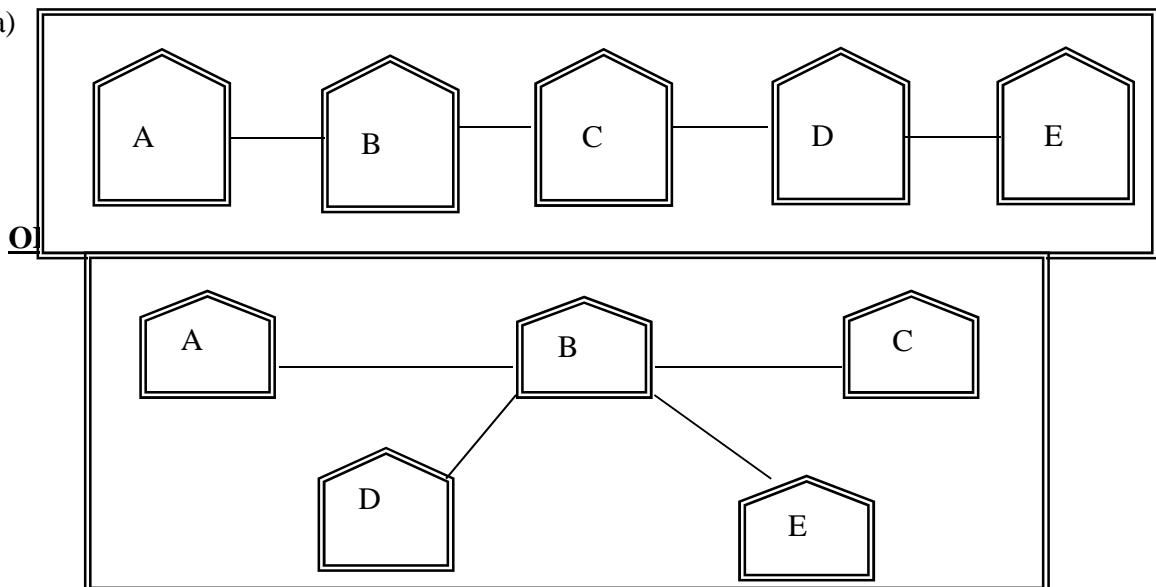


Distance between various buildings	
A to B	50 Mts
B to C	30 Mts
C to D	30 Mts
D to E	35 Mts
E to C	40 Mts
D to A	120 Mts
D to B	45 Mts
E to B	65 Mts

No of computers	
A	55
B	180
C	60
D	55
E	70

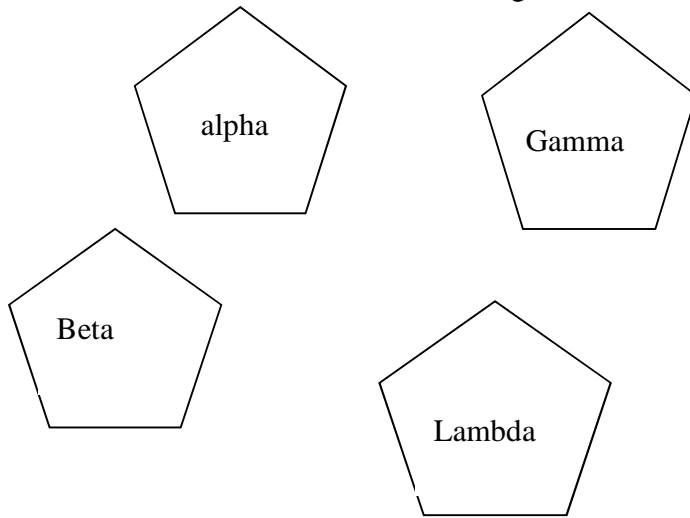
- (a) Suggest a possible cable layout for connecting the buildings.
 (b) Suggest the most suitable place to install the server of this organization.
 (c) Suggest the placement of the following devices with justification.
 (i.) Hub/Switch
 (ii.) Repeater
 (d) The company wants to link its head office in 'A' building to its Office in Sydney
 (i) Which type of transmission medium is appropriate for such a link?
 (ii) What type of network this connection result into?

Ans: (a)



- (b) The server should be installed in building B as it contains the maximum number of computers, thus decreasing the cabling cost for most of the computers as well as increasing the efficiency of the maximum computers in the network.
 (c) (i) A hub/switch each would be needed in all the buildings, to interconnect the group of cables from the different computers in each block.
 (ii) Since the cabling distances between buildings are not quite large, so no repeater is required.
 (d) (i) Internet (ii) WAN

Q. 6. Knowledge Supplement Organization has set up its new centre at Mangalore for its office and web based activities. It has four building as shown in the diagram below



Number of Computers:

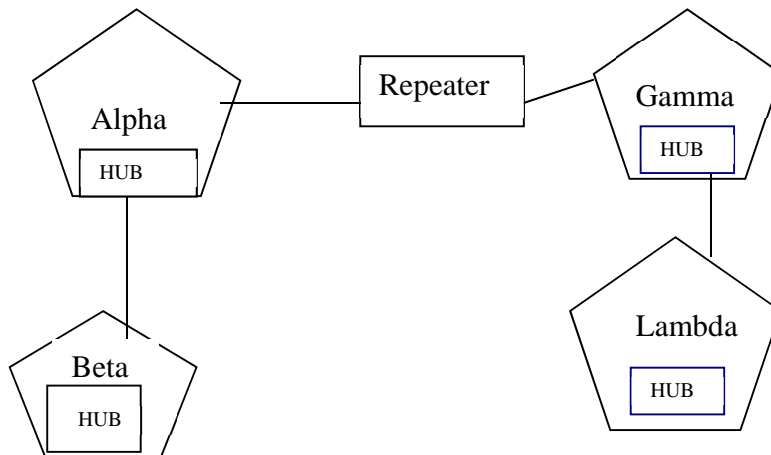
Alpha	25
Beta	50
Gamma	125
Lambda	10

Centre to Centre distance between various buildings

Alpha to Beta	50m
Beta to gamma	150m
Gamma to Lambda	25m
Alpha to Lambda	170m
Beta to Lambda	125m
Alpha to Gamma	90m

- Suggesting a cable layout of connection between building state with justification where Server, Repeater and hub will be placed.
- The organization is planning to link its front office situated in the city in a hilly region where cable connection is not feasible, suggest an economic way to connect it with reasonably high speed?

Ans: (i) The most suitable place to house the server of this organization would be building Gamma, as this building contains the maximum number of computers, thus decreasing the cabling cost for most of the computers as well as increasing the efficiency of the maximum computers in the network. Distance between alpha to gamma and beta to gamma is large so there repeater will require and hub is necessary for all premises because it is used in local networking.



(ii) The most economic way to connect it with a reasonable high speed would be to use radio wave transmission, as they are easy to install, can travel long distances, and penetrate buildings easily, so they are widely used for communication, both indoors and outdoors. Radio waves also have the advantage of being omni directional, which is they can travel in all the directions from the source, so that the transmitter and receiver do not have to be carefully aligned physically.

Q. 6. Software Development Company has set up its new center at Jaipur for its office and web based activities. It has 4 blocks of buildings as shown in the diagram below:



Center to center distances between various blocks

Block A to Block B	50 m
Block B to Block C	150 m
Block C to Block D	25 m
Block A to Block D	170 m
Block B to Block D	125 m
Block A to Block C	90 m

Number of Computers:	
Block A	25
Block B	50
Block C	125
Block D	10

- e1) Suggest a cable layout of connections between the blocks.
- e2) Suggest the most suitable place (i.e. block) to house the server of this company with a suitable reason.
- e3) Suggest the placement of the following devices with justification
 - (i) Repeater

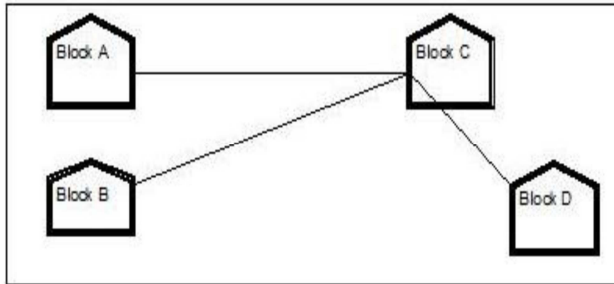
(ii) Hub/Switch

e4) the company is planning to link its front office situated in the city in a hilly region where cable connection is not feasible, suggest an economic way to connect it with reasonably high speed?

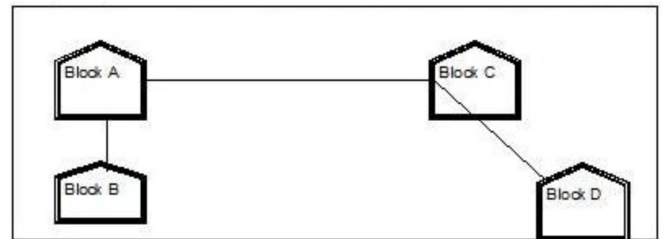
Ans:

(e1) (Any of the following option)

Layout Option 1



Layout Option 2



(e2) The most suitable place / block to house the server of this organization would be Block C, as this block contains the maximum number of computers, thus decreasing the cabling cost for most of the computers as well as increasing the efficiency of the maximum computers in the network.

(e3)

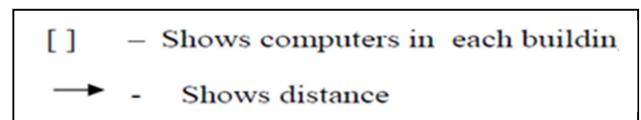
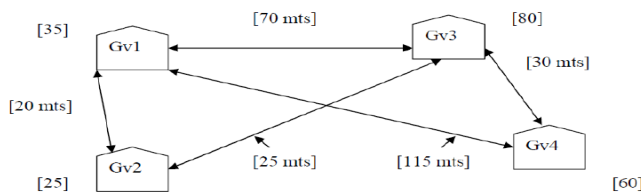
(i) For Layout 1, since the cabling distance between Blocks A and C, and that between B and C are quite large, so a repeater each would ideally be needed along their path to avoid loss of signals during the course of data flow in these routes.

For layout 2, since the distance between Blocks A and C is large so a repeater would ideally be placed in between this path.

(ii) In both the layouts, a hub/switch each would be needed in all the blocks, to Interconnect the group of cables from the different computers in each block.

(e4) The most economic way to connect it with a reasonable high speed would be to use radio wave transmission, as they are easy to install, can travel long distances, and penetrate buildings easily, so they are widely used for communication, both indoors and outdoors. Radio waves also have the advantage of being omni directional, which is they can travel in all the directions from the source, so that the transmitter and receiver do not have to be carefully aligned physically.

Q. 7. Ram Goods Ltd. has following four buildings in Ahmedabad city.



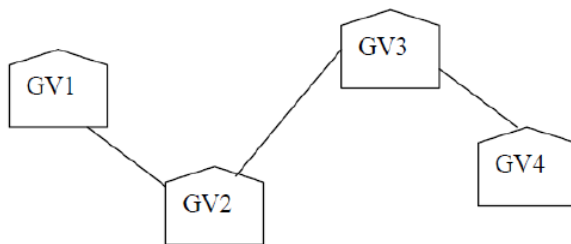
Computers in each building are networked but buildings are not networked so far. The company has now decided to connect building also.

(a) Suggest a cable layout for these buildings.

- (b) In each of the buildings, the management wants that each LAN segment gets a dedicated bandwidth i.e. bandwidth must not be shared. How can this be achieved?
- (c) The company also wants to make available shared Internet access for each of the buildings. How can this be achieved?
- (d) The company wants to link its head office in GV1 building to its another office in Japan.
 - (i) Which type of transmission medium is appropriate for such a link?
 - (ii) What type of network would this connection result into?

Ans:

(a) Total cable length required for this layout = 75 mts



- (b) To give dedicated bandwidth, the computers in each building should be connected via switches as switches offer dedicated bandwidths.
- (c) By installing routers in each building, shared internet access can be made Possible.
- (d) (i) Satellite as it can connect offices across globe.
(ii) WAN (Wide Area Network)

Q. 8.. Abhivandan Sanskriti in Udaipur is setting up the network between its different wings. There are 4 wings named as SENIOR(S), MIDDLE(M), JUNIOR(J) and OFFICE(O). Distance between the various wings are given below:

Wing O to Wing S	100m
Wing O to Wing M	200m
Wing O to Wing J	400m
Wing S to Wing M	300m
Wing S to Wing J	100m
Wing J to Wing M	450m

No. of Computers	
Wing O	10
Wing S	200
Wing M	100
Wing J	50

- (i) Suggest a suitable Topology for networking the computer of all wings.
- (ii) Name the wing where the server to be installed. Justify your answer.
- (iii) Suggest the placement of Hub/Switch in the network.
- (iv) Mention an economic technology to provide internet accessibility to all wings.

Ans:

- (i) Star or Bus or any other valid topology.
- (ii) Wing S, because maximum number of computers are located at Wing S.
- (iii) Hub/ Switch in all the wings.
- (iv) Coaxial cable/Modem/LAN/TCP-IP/Dialup/DSL/Leased Lines or any other valid technology.

SAMPLE PAPER - SET
MARKING SCHEME
COMPUTER SCIENCE [CODE-083]
CLASS - XII

Max Time : 3 hours

Max Marks : 70

1.	<p>(a) Write the prototype of a function named Percent, which takes an integer as value parameter and return a float type value. The parameter should have a default value 10.</p>	(2)
	<p>(b) Write the names of header files, which are NOT necessary to run the following program:</p> <pre> #include <iostream.h> #include <stdio.h> #include <string.h> #include <math.h> void main() { char STR[80]; gets(STR); puts(strrev(STR)); } </pre>	(1)
	<p>(c) Tarunaj has just started working as programmer in the JAGAT WORLD SOFTWARE company. In the company, he has got his first assignment to develop a small C++ module to find the biggest number out of a given set of numbers stored in a one dimensional array. Somehow he has committed a few logical mistakes while writing this code and so he is not getting the desired result from the code. Find out the mistakes and correct this C++ code so that it provides the desired result (do not add any new statement in the code). Underline each correction made:</p> <pre> int BIGFIND(int ARR[],int Size) { int BIG=ARR[1]; //Statement 1 for (int C=2;C<Size;C++) //Statement 2 if (ARR[C]<BIG) //Statement 3 ARR[C]=BIG; //Statement 4 return BIG; //Statement 5 } </pre>	(2)
	<p>(d) Find output of the following program segment:</p> <pre> int A[][3] = {{1,2,3}, {5,6,7}}; for (int i = 1; i<2; i++) for (int j = 0; j<3; j++) cout<<A[i][j]<<"*\n"; </pre>	(2)
	<p>(e) Find output of the following program segment:</p> <pre> int a = 3; void demo(int x, int y, int &z) { a += x+y; z = a+y; y += x; cout<<x<<' '<<y<<' '<<z<<endl; } void main() </pre>	(3)

	<pre> { int a = 2, b = 5; demo (::a, a, b); demo (::a, a, b); } </pre>	
	(f) Write a function in C++ to accept two integers as parameters and returns the greater of these numbers.	(2)
2.	(a) What do you understand by Data Encapsulation and Data Hiding? Also, give a suitable C++ code to illustrate both.	(2)
	(b) What is constructor overloading? Give an example to illustrate the same.	(2)
	<p>(c) Define a class HandSet in C++ with following description:</p> <p>Private members: Make- of type string Model- of type string Price- of type long int Rating- of type char</p> <p>Public Members: Function Read_Data to read an object of HandSet type. Function Display() to display the details of an object of HandSet type. Function RetPrice() to return the value of Price of an object of HandSet type.</p>	(4)
	<p>(d) Consider the following class counter:</p> <pre> class counter { protected : unsigned int count; public : counter() { count = 0; } void inc_count() { count++; } int get_count() { return count; } }; </pre> <p>Write code in C++ to publically derive another class new_counter from class counter. Class new_counter should have the following additional function members in the public visibility mode:</p> <p>(i) A parameterized constructor to initialize the value of count to the value of parameter. (ii) dec_count() to decrease the value of data member count by 1. (iii) Reset() to set the value of data member count to 0.</p>	(4)
3.	<p>(a) Write a function TRANSFER(int A[], int B[], int Size) in C++ to copy the elements of array A into array B in such a way that all the negative elements of A appear in the beginning of B, followed by all the positive elements, followed by all the zeroes maintaining their respective orders in array A. For example:</p> <p>If the contents of array A are: 7, -23, 3, 0, -8, -3, 4, 0 The contents of array B should be -23, -8, -3, 7, 3, 4, 0</p>	(3)
	(b) Each element of the array A[8][6] is stored using 4 bytes of memory. If the element A[2][4] is stored at location 936, find the address of A[5][1]. Assume that the array	(3)

	is stored column-wise.									
	(c) Write a function in C++ to perform Insert operation in a circular Queue containing Player's information (represented with the help of an array of structure PLAYER).	(4)								
	<pre> struct PLAYER { long PID; //Player ID char Pname[20]; //Player Name }; </pre>									
	(d) Write a function TRANSFORM(int A[4][3]) in C++ to swap the elements of the first column with the corresponding elements of last column of array A.	(2)								
	(e) Convert the expression $(A-5)*6+(10/B)/2$ to corresponding postfix expression. Also show the status of operator stack after each step.	(2)								
4.	(a) A binary file "Students.dat" contains data of 10 students where each student's data is an object of the following class: <pre> class Student { int Rno;char Name [20] ; public: void EnterData() {cin>>Rno; cin.getline(Name,20) ; void ShowData() {cout<<Rno<<" - "<<Name<<endl; } }; </pre> <p>With reference to this information, write output of the following program segment:</p> <pre> ifstream File; Student S; File.open("STUDENTS.DAT",ios::binary ios::in) ; File.seekg(0, ios::end) ; Cout<<File.tellg() ; </pre>	(1)								
	(b) Write a function in C++ to count the number of lines starting with a digit in a text file "DIARY.TXT".	(2)								
	(c) Given a binary file "STUDENT.DAT", containing records of the following class Student type: <pre> class student { char S_admno[10]; //Admission no. of student char S_Name[20]; //Name of student int Percentage; //Marks percentage of student public: void EnterData() { gets(S_admno); gets(S_Name); cin>>Percentage; } void DisplayData() { cout<<setw(12)<<S_admno; cout<<setw(32)<<S_Name; cout<<setw(3)<<Percentage<<endl; } int Ret_Per() {return Percentage;} }; </pre> <p>Write a function in C++ that would read contents of the file "STUDENT.DAT" and display the details of those students whose percentage is above 75.</p>	(3)								
5.	(a) Observe the following Table and answer the parts (i) and (ii) accordingly <p style="text-align: center;">Table: MEMBER</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Mno</th> <th>Name</th> <th>Qty</th> <th>PurchaseDate</th> </tr> </thead> <tbody> <tr> <td>101</td> <td>Pen</td> <td>102</td> <td>12-12-2011</td> </tr> </tbody> </table>	Mno	Name	Qty	PurchaseDate	101	Pen	102	12-12-2011	(2)
Mno	Name	Qty	PurchaseDate							
101	Pen	102	12-12-2011							

102	Pencil	201	21-02-2012
102	Eraser	90	09-08-2010
109	Sharpener	90	31-08-2012
113	Clips	900	08-08-2011

(i) In the above table, can we take Mno as Primary Key? (Answer as [YES/NO] only).
Justify your answer with a valid reason.

(ii) What is the degree and the cardinality of the above table?

Consider the following tables SUBJECT and TEACHER and answer (b), (c), (d) and (e) parts of this question:

Table: SUBJECT

Code	Title	Marks_Theory	Marks_Prac
301	English	100	0
041	Maths	100	0
083	Computer Sc.	70	30
042	Physics	70	30
043	Chemistry	70	30

Table: TEACHER

TCode	Name	Sub_Code
1	P. Jain	301
2	R. Nagpal	301
3	Supatra	041
4	Shabnam	083
5	Rashika	042
6	Vidushi	041
7	Yash	043

(b) Write SQL commands for the following statements:

(4)

- To display the names of all the subjects for which practical marks are 0.
- To display the total number of teachers in each subject separately.
- To display the names of all the teachers in the ascending order of the Sub_Code.
- To display each subject's details along with Total_Marks in each subject from the table SUBJECT. (Total_Marks = Marks_Theory + Marks_Practical).

(c) Write SQL statement to display each teacher's name along with his/her respective subject name from the tables TEACHER and SUBJECT.

(2)

(d) Give the output of the following SQL queries:

(1)

- `SELECT DISTINCT (Marks_Theory) from SUBJECT;`
- `SELECT TCode, Name from Teacher where Sub_Code like '0%';`

(e) Identify primary keys of the tables SUBJECT and TEACHER.

(1)

6. (a) State the dual of the absorption law $X+X.Y = X$ and prove it algebraically.

(2)

(b) Draw the logic diagram for the Boolean expression $X.(Y'+Z)$ using basic logic gates.

(2)

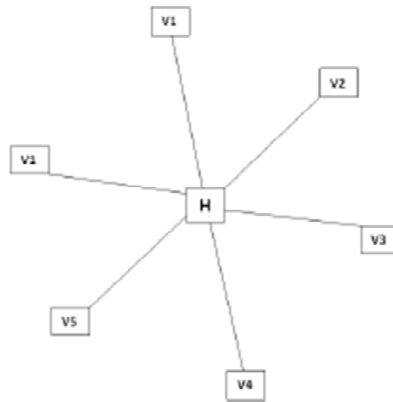
(c) Write the SOP form of the Boolean function $F(P,Q,R) = \Sigma(0,2,3,5)$.

(1)

(d) Find the simplified expression for the following Boolean function using Karnaugh's map: $F(A, B, C, D) = \Sigma(0,1,2,4,5,6,8,9,10)$

(3)

7. (a) To provide telemedicine facility in a hilly state, a computer network is to be setup to connect hospitals in 6 small villages (V1, V2, . . . , V6) to the base hospital (H) in the state capital. This is shown in the following diagram.



No village is more than 20km away from the state capital.

Imagine yourself as a computer consultant for this project and answer the following questions with justification:

(i) Out of the following what kind of link should be provided to setup this network: (i) Microwave link, (ii) Radio Link, (iii) Wired link?	(2)
(ii) What kind of network will be formed: LAN, MAN, or WAN?	(1)
(iii) Many times doctors at village hospital will have to consult senior doctors at the base hospital. For this purpose, how should they contact them: using email, SMS, telephone, or video conference?	(1)
(b) Out of SMTP and POP3 which protocol is used to receive emails?	(1)
(c) What are cookies in the context of computer networks?	(1)
(d) Rajeshwari is trying for on-line subscription to a magazine. For this she has filled in a form on the magazine's web site. When she clicks submit button she gets a message that she has left e-mail field empty and she must fill it. For such checking which type of script is generally executed - client-side script or server-side script?	(1)
(e) Mention any one difference between freeware and free software.	(1)

Privious Year (2014) CBSE Question Paper

Series : OSR/1

Code No. 91/1

Roll No.

--	--	--	--	--	--	--	--

Candidates must write the Code on the title page of the answer-book.

- Please check that this question paper contains 11 printed pages.
- Code number given on the right hand side of the question paper should be written on the title page of the answer-book by the candidate.
- Please check that this question paper contains 7 questions.
- **Please write down the Serial Number of the question before attempting it.**
- 15 minutes time has been allotted to read this question paper. The question paper will be distributed at 10.15 a.m. From 10.15 a.m. to 10.30 a.m., the students will read the question paper only and will not write any answer on the answer-book during this period.

COMPUTER SCIENCE

Time allowed : 3 hours]

[Maximum Marks : 70

Instructions : (i) All questions are compulsory.
(ii) Programming Language : C++

1. (a) What is the difference between actual and formal parameter ? Give a suitable example to illustrate using a C++ code. 2
- (b) Observe the following C++ code and write the name(s) of the header file(s), which will be essentially required to run it in a C++ compiler : 1

```
void main( )  
{  
    char Text [20] , C;  
    cin>>Text;  
    C=tolower(Text [0]);  
    Cout<<C<<" is the first char of " <<Text<<endl;  
}
```

- (c) Rewrite the following C++ code after removing all the syntax error(s), if present in the code. Make sure that you underline each correction done by you in the code.

2

Important Note :

- Assume that all the required header files are already included, which are essential to run this code.
- The corrections made by you do not change the logic of the program.

```
typedef char [50] STRING;
void main( )
{
    City STRING;
    gets(City);
    cout<<City[0]<<' \ t<<City[2];
    cout<<City<<endl;
}
```

- (d) Obtain the output from the following C++ program as expected to appear on the screen after its execution.

2

Important Note :

- All the desired header files are already included in the code, which are required to run the code.

```
void main( )
{
    Char *String="SARGAM";
    int *Ptr, A[]={1,5,7,9};
    Ptr=A;
    cout<<*Ptr<<String<<endl;
    String++;
    Ptr+=3;
    cout<<*Ptr<<String<<endl;
}
```

- (e) Obtain the output of the following C++ program, which will appear on the screen after its execution.

Important Note :

3

- All the desired header files are already included in the code, which are required to run the code.

Class Player

```
{
    int Score,Level;
    char Game;
```



```

public:
    Player(char GGame='A')
    {Score=0; Level=1; Game=GGame;}
    void Start(int SC);
    void Next();
    void Disp()
    {
        cout<<Game<<"@"<<Level<<endl;
        cout<<Score<<endl;
    }
};
void main()
{
    Player P,Q('B');
    P.Disp();
    Q.Start(75);
    Q.Next();
    P.Start(120);
    Q.Disp();
    P.Disp();
}
void Player::Next()
{
    Game=(Game=='A')?'B':'A';
}
void Player::Start(int SC)
{
    Score+=SC;
    If(Score>=100)
        Level=3;
    else if(Score>=50)
        Level=2;
    else
        Level=1;
}

```

- (f) Read the following C++ code carefully and find out, which out of the given options (i) to (iv) are the expected correct output(s) of it. Also, write the maximum and minimum value that can be assigned to the variable Start used in the code:

2

```

void main()
{
    int Guess[4]={200,150,20,250};
    int Start=random(2)+2;
    for (int C=Start;C<4;C++)
        cout<<Guess[C]<<"#";
}

```

- | | |
|--------------|-------------------|
| (i) 200#150# | (iii) 150#20#250# |
| (ii) 150#20# | (iv) 20#250# |

- (a) Write 4 characteristics of a constructor function used in a class. 2
 (b) Answer the questions (i) and (ii) after going through the following class : 2

```

class Health
{
    int PId, DId;
public :
    Health(int PPID);           //Function 1
    Health();                   //Function 2
    Health (Health &H);        //Function 3
    void Entry();              //Function 4
    void Display();            //Function 5
};
void main ( )
{
    Health H(20);               //Statement 1
}
  
```

- (i) Which of the function out of Function 1, 2, 3, 4 or 5 will get executed when the Statement 1 is executed in the above code ?
 (ii) Write a statement to declare a new object G with reference to already existing object H using Function 3.
- (c) Define a class CABS in C++ with the following specification: 4

Data Members

- CNo - to store Cab No
- Type - to store a character 'A', 'B' or 'C' as City Type
- PKM - to store per Kilo Meter charges
- Dist - to store Distance travelled (in KM)

Member Functions

- A constructor function to initialize Type as 'A' and CNo as '1111'
- A function Charges() to assign PKM as per the following table:

Type	PKM
A	25
B	20
C	15

- A function Register() to allow administrator to enter the values for CNo and Type. Also, this function should call Charges() to assign PKM Charges.
 - A function ShowCab() to allow user to enter the value of Distance and display CNo, Type, PKM, PKM*Distance (as Amount) on screen
- (d) Consider the following C++ code and answer the questions from (i) to (iv) : 4

```

Class Campus
{
    long Id;
    char City[20];
protected:
    char Country[20];
}
  
```

```

public :
    Campus ();
    void Register ();
    void Display ();
};
class Dept: private Campus
{
    long DCode [10];
    char HOD [20];
protected :
    double Budget;
public:
    Dept ();
    void Enter ();
    void Show ();
};
class Applicant: public Dept
{
    long RegNo;
    char Name [20];
public:
    Applicant ();
    void Enroll ();
    void View ();
};

```

- (i) Which type of Inheritance is shown in the above example ?
- (ii) Write the names of those member functions, which are directly accessed from the objects of class Applicant.
- (iii) Write the names of those data members, which can be directly accessible from the member functions of class Applicant.
- (iv) Is it possible to directly call function Display() of class University from an object of class Dept ? (Answer as Yes or No).

- (a) Write code for a function oddEven (int s [], int N) in C++, to add 5 in all the odd values and 10 in all the even values of the array S. 3

Example: If the original content of the array S is

S[0]	S[1]	S[2]	S[3]	S[4]
50	11	19	24	28

The modified content will be :

S[0]	S[1]	S[2]	S[3]	S[4]
60	16	24	34	38

- (b) An array T[25][20] is stored along the row in the memory with each element requiring 2 bytes of storage. If the base address of array T is 42000, find out the location of T[10][15]. Also, find the total number of elements present in this array. 3

- (c) Write a user-defined function `SumLast3(int A[][4], int N, int M)` in C++ to find and display the sum of all the values, which are ending with 3 (i.e., units place is 3). For example if the content of array is :

2

33	13	92
99	3	12

The output should be

49

- (d) Evaluate the following postfix expression. Show the status of stack after execution of each operation separately :

2

F, T, NOT, AND, F, OR, T, AND

- (e) Write a function `POPBOOK()` in C++ to perform delete operation from a Dynamic Stack, which contains Bno and Title. Consider the following definition of `NODE`, while writing your C++ code.

4

```
struct NODE
{
    int Bno;
    char Title[20];
    NODE *Link;
};
```

4. (a) Fill in the blanks marked as Statement 1 and Statement 2, in the program segment given below with appropriate functions for the required task.

1

```
class Medical
{
    int RNo;                //Representative Code
    char Name[20];         //Representative Name
    char Mobile[12];       //Representative Mobile
public:
    void Input();//Function to enter all details
    void Show();//Function to display all details
    int RRno(){return RNo;}
    void ChangeMobile();//Function to change Mobile
    {
        cout<< "Changed Mobile:";
        gets(Mobile);
    }
}
```

```

};
void RepUpdate()
{
    fstream F;
    F.open("REP.DAT", ios::binary|ios::in|ios::out);
    int Change=0;
    int URno;
    cout<<"Rno(Rep No-to update Mobile):";
    cin>>URno;
    Medical M;
    while(!Change && F.read((char*)&M, sizeof(M)))
    {
        if (M.Rrno() == URno)
        {
            //Statement 1:To call the function to change Mobile No.
            _____;
            //Statement 2:To reposition file pointer to re-write
            //the updated object back in the file
            _____;
            F.write((char*)&M, sizeof(M));
            Change++;
        }
    }
    if (Change)
        cout<<"Mobile Changed for Rep "<<URno<<endl;
    else
        cout<<"Rep not in the Medical"<<endl;
    F.close();
}

```

- (b) Write a function EUCount () in C++, which should read each character of a text file IMP.TXT, should count and display the occurrence of alphabets E and U (including small cases e and u too). 2

Example :

If the file content is as follows:

Updated information

is simplified by official websites.

The EUCount() function should display the output as

E:4

U:1

- (c) Assuming the class GAMES as declared below, write a functions in C++ to read the objects of GAMES from binary file GAMES.DAT and display those details of those GAMES, which are meant for children of AgeRange "8 to 13". 3

```
Class GAMES
{
    int GameCode;
    char GameName [10];
    char AgeRange;
public :
    void Enter()
    {
        cin>>GameCode;
        gets (GameName);
        gets (AgeRange);
    }
    void Display()
    {
        cout<<GameCode<<" : " <<GameName<<endl;
        cout<<AgeRange<<endl;
    }
    char* AgeR () {return AgeRange};
};
```

5. (a) Explain the concept of Union between two tables, with the help of appropriate example. 2

NOTE :

Answer the questions (b) and (c) on the basis of the
following tables **STORE** and **ITEM**

Table: STORE

SNo	SName	Area
S01	ABC Computronics	GK II
S02	All Infotech Media	CP
S03	Tech Shoppe	Nehru Place
S04	Geeks Tecno Soft	Nehru Place
S05	Hitech Tech Store	CP

Table : ITEM

INo	IName	Price	SNo
T01	Mother Board	12000	S01
T02	Hard Disk	5000	S01
T03	Keyboard	500	S02
T04	Mouse	300	S01
T05	Mother Board	13000	S02
T06	Key Board	400	S03
T07	LCD	6000	S04
T08	LCD	5500	S05
T09	Mouse	350	S05
T10	Hard Disk	4500	S03

(b) Write the SQL queries (1 to 4) 4

- (1) To display IName and Price of all the Items in ascending order of their Price.
- (2) To display SNo and SName of all Stores located in CP
- (3) To display Minimum and Maximum Price of each IName from the table Item.
- (4) To display IName, Price of all items and their respective SName where they are available.

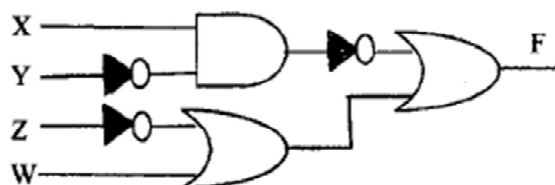
(c) Write the output of the following SQL commands (1 to 4) : 2

- (1) SELECT DISTINCT INAME FROM ITEM WHERE PRICE >= 5000;
- (2) SELECT AREA, COUNT(*) FROM STORE GROUP BY AREA;
- (3) SELECT COUNT(DISTINCT AREA) FROM STORE;
- (4) SELECT INAME, PRICE * 0.05 DISCOUNT FROM ITEM WHERE SNO IN ('S02', 'S03');

6. (a) Name the law shown below and verify it using a truth table. 2

$$A+B \cdot C = (A+B) \cdot (A+C)$$

(b) Obtain the Boolean Expression for the logic circuit shown below : 2



- (c) Write the Sum of Product form of the function $F(P,Q,R)$ for the following truth table representation of F :

1

P	Q	R	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

- (d) Obtain the minimal form for the following Boolean expression using Karnaugh's Map.

3

$$F(A, B, C, D) = \Sigma (1, 4, 5, 9, 11, 12, 13, 15)$$

7. (a) Write one characteristics each for 2G and 3G Mobile Technologies.

1

- (b) What is the difference between Video Conferencing and Chat ?

1

- (c) Expand the following :

1

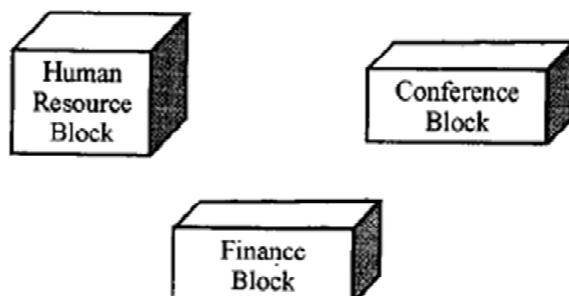
- GPRS
- CDMA

- (d) Which type of network (out of LAN, PAN and MAN) is formed, when you connect two mobiles using Bluetooth to transfer a picture file.

1

- (e) Trine Tech Corporation (TTC) is a professional consultancy company. The company is planning to set up their new offices in India with its hub at Hyderabad. As a network adviser, you have to understand their requirement and suggest them the best available solutions. Their queries are mentioned as (i) to (iv) below.

Physical Locations of the blocks of TTC



Block to Block distances (in Mtrs.)

Block (From)	Block (To)	Distance
Human Resource	Conference	110
Human Resource	Finance	40
Conference	Finance	80

Expected Number of Computers to be installed in each block

Block	Computers
Human Resource	25
Finance	120
Conference	90

- (i) What will the most appropriate block, where TTC should plan to install their server ? 1
- (ii) Draw a block to block cable layout to connect all the buildings in the most appropriate manner for efficient communication. 1
- (iii) What will be the best possible connectivity out of the following, you will suggest to connect the new setup of offices in Bangalore with its London based office. 1
- Satellite Link
 - Infrared
 - Ethernet Cable
- (iv) Which of the following device will be suggested by you to connect each computer in each of the buildings ? 1
- Switch
 - Modem
 - Gateway
- (f) Write names of any two popular Open Source Software, which are used as operating system. 1
- (g) Write any two important characteristics of Cloud Computing. 1