COMPUTER SCIENCE C++(083)

ASSIGNMENT BOOKLET

CLASS XII

Academic Session: 2016-17

INDEX

Class XII

Computer Science Syllabus
Class XII

March

Object Oriented Programming
Function Overloading

April

Classes and Objects
Constructors and Destructors
Inheritance: Extending Classes

May

Arrays
Project Allocation

July

Pointers
Data File Handling

August

Linked Lists, Stacks & Queues

September

First Term Examination
Boolean Algebra

October

Concepts of Networking
Structured Query Language

November

Revision
Project and File Submission

December

Second Term Examination

Unit-1
Object Oriented Programming In C++

Handout I– OOPS Concepts

Programming Paradigm: it defines the methodology of designing and implementing programs using the key features and building blocks of a programming language.

Procedural Programming: In procedural programming paradigm, the emphasis is on doing things i.e., the procedure or the algorithm. The data takes the back seat in procedural programming paradigm. Also, this paradigm does not model real world well.

Object Oriented Programming: The object oriented programming paradigm models the real world well and overcomes the shortcomings of procedural paradigm. It views a problem in terms of objects and thus emphasizes on both procedures as well as data. It follows a bottom up approach in program design and emphasizes on safety and security of data.

Important Concepts in OOPS: Data Abstraction:

It refers to the act of representing essential features without including the background details. Example: For driving , only accelerator, clutch and brake controls need to be learnt rather than working of engine and other details.

Data Encapsulation:

It means wrapping up data and associated functions into one single unit called class.

A class groups its members into three sections: public, private and protected, where private and protected members remain hidden from outside world and thereby helps in implementing data hiding.

Modularity:

The act of partitioning a complex program into simpler fragments called modules is called as modularity.

It reduces the complexity to some degree.

It creates a number of well defined boundaries within the program.

Inheritance:

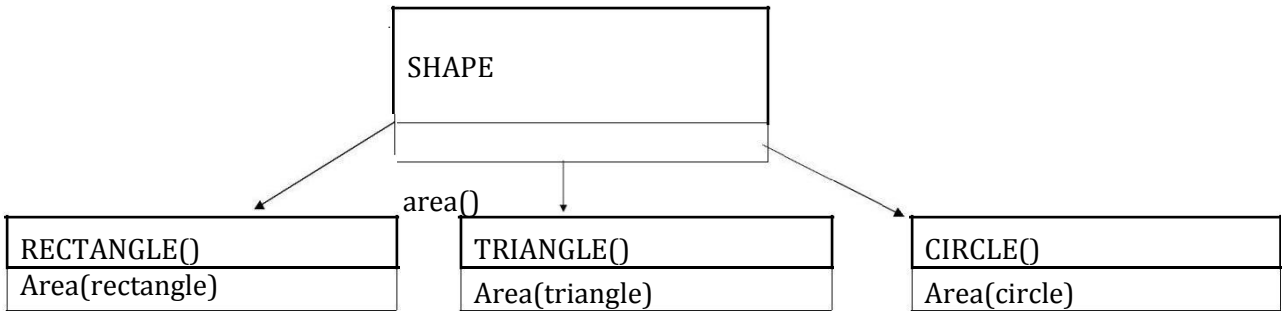
Inheritance is the process of forming a new class from an existing class or base class. The base class is also known as parent class or super class.

Derived class is also known as a child class or sub class. Inheritance helps in reusability of code , thus reducing the overall size of the program.

Polymorphism:

Poly means many and **morphs** mean form, so polymorphism means one name multiple forms. It is the ability for a message or data to be processed in more than one form.

C++ implements Polymorphism through Function Overloading, Operator overloading and Virtual functions.



Advantages of OOPS:

1. Reusability of code
2. Ease of comprehension.
3. Ease of fabrication and maintenance.
4. Easy redesign and extension.

Handout II –Implementing Polymorphism in C++ through Function Overloading

C++ allows you to specify more than one definition for a **function** name in the same scope, which is called **function overloading**.

An overloaded declaration is a declaration that had been declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and obviously different definition (implementation).

When you call an overloaded **function**, the compiler determines the most appropriate definition to use by comparing the argument types you used to call the function with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function is called **overload resolution**.

You can have multiple definitions for the same function name in the same scope. The definition of the function must differ from each other by the types and/or the number of arguments in the argument list. **You can not overload function declarations that differ only by return type**.

Following is the example where same function **print()** is being used to print different data types:

```
#include <iostream.h>
void print(int i) {
      cout << "Printing int: " << i << endl;                }
void print(double f) {
      cout << "Printing float: " << f << endl;              }
void print(char* c) {
      cout << "Printing character: " << c << endl;        }
int main(void)
{
  print(5);              // Call print to print integer
  print(500.263);  // Call print to print float
  print("Hello C++");              // Call print to print character
   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Printing int: 5 Printing
float: 500.263
Printing character: Hello C++
```

Handout III – Classes and Objects in C++

Classes:

The classes are the most important feature of C++ that leads to Object Oriented programming. Class is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating instance of that class.

The variables inside class definition are called as data members and the functions are called member functions.

For example: Class of birds, all birds can fly and they all have wings and beaks. So here flying is a behavior and wings and beaks are part of their characteristics. And there are many different birds in this class with different names but they all posses this behavior and characteristics.

Similarly, class is just a blue print, which declares and defines characteristics and behavior, namely data members and member functions respectively. And all objects of this class will share these characteristics and behavior.

A **Class** is a group of similar objects that share common characteristics and relationships. It acts as a blueprint for defining objects.

It is a collection of variables, often of different types and its associated functions.

Class just binds data and its associated functions under one unit there by enforcing encapsulation.

Classes define types of data structures and the functions that operate on those data structures.

Declaration/Definition:

A class definition starts with the keyword class followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations.

```
class class_name {
                    access_specifier_1: member1;
                    access_specifier_2: member2;
                    ...
                    } object_names;
```

Where class_name is a valid identifier for the class, object_names is an optional list of names for objects of this class. The body of the declaration can contain members that can either be data or function declarations, and optionally access specifiers.

Note: the default access specifier is private.

```
Example: class Box {
                        int a;
                    public:
                        double length; // Length of a box
                        double breadth; // Breadth of a box
                        double height; // Height of a box
};
```

Access specifiers in Classes:

Access specifiers are used to identify access rights for the data and member functions of the class. There are three main types of access specifiers in C++ programming language: private,

public, protected.

Private	Class members declared as private can be used only by member functions and friends (classes or functions) of the class.
Protected	Class members declared as protected can be used by member functions and friends (classes or functions) of the class. Additionally, they can be used by classes derived from the class.
Public	Class members declared as public can be used by any function.

Why Access Specifiers?

Access control helps prevent you from using objects in ways they were not intended to be used. Thus it helps in implementing data hiding and data abstraction.

Objects:

Class is mere a blueprint or a template. No storage is assigned when we define a class. Objects are instances of class, which holds the data variables declared in class and the member functions work on these class objects. Objects are basic run time entities in an object oriented system.

An **object** is an identifiable entity with some characteristics and behavior.

For example: Dogs have characteristics (name, color, breed) and behavior (barking, fetching, wagging tail). Cars also have characteristics (number of doors, number of seats, type of engine, fuel type, average performance) and behavior (moving forward, moving backward, turning, changing gear, applying brakes). Identifying the characteristics and behavior for real-world objects is a great way to begin thinking in terms of object-oriented programming. These real-world observations all translate into the world of object-oriented programming.

Software objects are conceptually similar to real-world objects: they too consist of state and related behavior. An object stores its state in fields (variables in some programming languages) and exposes its behavior through functions.

Creating object / defining the object of a class:

The general syntax of defining the object of a class is:-

Class_name object_name;

In C++, a class variable is known as an object. The declaration of an object is similar to that of a variable of any data type. The members of a class are accessed or referenced using object of a class.

```
Box Box1;          //Declare Box1 of type Box
Box Box2;          //Declare Box2 of type Box
```

Both of the objects Box1 and Box2 will have their own copy of data members.

Accessing / referencing members of a class:

All member of a class are private by default. Private member can be accessed only by the function of the class itself. Public member of a class can be accessed through any object of the class. They are accessed or called using object of that class with the help of dot operator (.).

The general syntax for accessing data member of a class is:-

Object_name.Data_member=value;

The general syntax for accessing member function of a class is:-

Object_name. Function_name (actual arguments);

The dot ('. ') used above is called the **dot operator or class member access operator**. The dot operator is used to connect the object and the member function. The private data of a class can be accessed only through the member function of that class.

Class methods definitions (Defining the member functions):

Member functions can be defined in two places:-

Outside the class definition

The member functions of a class can be defined outside the class definitions. It is only declared inside the class but defined outside the class. The general form of member function definition outside the class definition is:

Return_type Class_name:: function_name (argument list)

{
Function body
}

Where symbol :: is a **scope resolution operator**.

The **scope resolution operator (::)** specifies the class to which the member being declared belongs, granting exactly the same scope properties as if this function definition was directly included within the class definition.

```
class sum
{ int A, B, Total;
public:
        void getdata ();
        void display ();
};
void sum:: getdata ()          // Function definition outside class definition Use of :: operator
{ cout<<" \n enter the value of A and B";
cin>>A>>B; }
void sum:: display ()          // Function definition outside class definition Use of :: operator
{ Total =A+B;
cout<<"\n the sum of A and B="<<Total; }
```

Inside the class definition

The member function of a class can be declared and defined inside the class definition.

```
class sum
{ int A, B, Total;
public:
        void getdata ()
                { cout< "\n enter the value of A and B"; cin>>A>>B;} void
        display ()
                { total = A+B;cout<<"\n the sum of A and B="<<total;}
};
```

Passing an Object as an Argument to a member function

```
/*C++ PROGRAM TO PASS OBJECT AS AN ARGUMEMT. The program adds the two heights given
in feet and inches. */
#include<iostream.h>
#include<conio.h>
 class height
{ int feet, inches;
public:
        void getht(int f,int i)
                {  feet=f;  inches=i;
        } void putheight()
                {cout<< "\nHeight is:"<< feet<< "feet\t"<< inches << "inches"<< endl;}
        void sum(height a, height b)
                { height n;
                        n.feet = a.feet + b.feet;
                        n.inches = a.inches + b.inches;
                        if(n.inches >=12)
                        { n.feet++;
                                n.inches = n.inches -12;
                        }
                 cout<< endl<< "Height is: "<< n.feet<< " feet and "<< n.inches<< "inches" <<endl;
                 }
};
void main()
 { height h,a,d;
        clrscr();
        h.getht(6,5);
        a.getht(2,7);
        h.putheight();
        a.putheight();
        d.sum(h,a);
        getch(); }
/**********OUTPUT***********
Height is: 6 feet 5 inches
Height is: 2 feet 7 inches
Height is: 9 feet and 0 inches
```

What is the difference between struct and classes in C++:

In C++, a *structure* is a class defined with the struct keyword. Its members and base classes are public by default. A class defined with the class keyword has private members and base classes by default. This is the only difference between struct and classes in C++.

Inline Functions

Inline functions definition starts with keyword inline.

The compiler replaces the function call statement with the function code itself(expansion) and then compiles the entire code.

They run little faster than normal functions as function calling overheads are saved.

A function can be declared inline by placing the keyword inline before it.

```
inline void square(int a)
{     cout<<a*a;  }
void main()
{               square(4);
                square(8);
}
```

These two statements are translated to: {cout<<4*4;} and {cout<<8*8;} respectively as the function is inline, its call is replaced by its body.

Assignment No. 1

Q 1. Define classes. Give one example.

Q 2. What do you understand by instance of a class. Give one example.

Q 3. What is the need of object oriented programming?

Q 4. Differentiate between Procedural programming and object oriented programming.

Q 5. Define data encapsulation. How is it implemented in C++?

Q 6. Define data hiding. How is it implemented in C++?

Q 7. Define data abstraction. How is it implemented in C++?

Q 8. Define polymorphism. How is it implemented in C++?

Q 9. Differentiate between public, private and protected data members.

Q 10. What do you understand by access specifiers?

Q 11. Differentiate between Inline Functions and Non Inline Functions. Give a suitable example using C++ code to illustrate the same.

Q 12. Define a class in C++ with the following description:

- A data member ***Flight number*** of type integer
- A data member ***Destination*** of type string
- A data member ***Distance*** of type float
- A data member ***Fuel*** of type float
- A member function **CALFUEL( )** to calculate the value of fuel as per the flowing criteria.

Distance	**Fuel**
<= 1000	500
More than 1000 and <= 2000	1100
More than 2000	2200

Public Members

- A function **FEEDINFO( )** to allow to enter values for flight number, destination, Distance & call function **CALFUEL( )** to calculate the quantity of fuel
- A function **SHOWINFO( )** to allow user to view the content of all the data members

Q 13. Given the following code fragment:

```
#include<iostream.h>
# include<conio.h>
class X
{ int x,y;
        void count(void);
   public :
            int a,b;
            void getval(int, int);
            void prval(void);
 };
: //X's member function's definition
```

```
X O1;
void main( )
{  class Y
    {
        int i,j; void
        abc( );
      public : float
        k; void
        get( );
        void prin( );
    };
Y O2;
X O3;
}  //end of main( )

void func1( )
{  X O4;
   :    }
void func2(
) { : }
```

Which all members (data & functions) of classes X and Y can be accessed by main(), func1(), func2( ) **?**

Q 14. A class '**time**' has the following
members: Data Members:
Hour of type int
Minute of type int
Member Function: Readtime(
int h, int m);
Showtime( );
Addtime(time T1, time T2);
Write a program using classes to input two different objects FT and ST, print their sum(assuming 24 hours clock time) e.g.,
INPUT :
FT= 6 hrs 35 minutes
ST= 3 hrs 45 minutes
OUTPUT :
T = FT+ST
= 10 hrs 20 minitues.

Q 15. What are static class members? Explain with a suitable example.

Q 16. What are nested classes?

Handout IV: Constructors and Destructors

CONSTRUCTORS :

A member function with the same as its class is called Constructor and it is used to initialize the object of that class with a legal initial value. It is fired automatically as soon as an object of the given class is created. It need not be called explicitly. But it must be placed inside the public section of the class definition.

Example :

```
class student
{ int rollno; float
        marks;
  public:
        student( )   //Constructor
        { rollno=0; marks=0.0;
                }
                   //other public members
};
```

TYPES OF CONSRUCTORS:

1. Default Constructor:

A constructor that accepts no parameter is called the Default Constructor. If you don't declare a constructor or a destructor, the compiler makes one for you. The default constructor and destructor take no arguments and do nothing. Default constructor provided by the compiler initializes the object data members to default value.

```
class Cube
{           int side;
        public:
           Cube()                //default constructor
            { side=10; }
};

int main()
{ Cube c;
    cout << c.side;
}
OUTPUT: 10
```

2. Parameterized Constructors:

A constructor that accepts parameters is known as parameterized Constructors, also called as Regular Constructors. Using this Constructor you can provide different values to data members of the objects, by passing the appropriate values as argument.

```
class Cube
{     int     side;
        public:
           Cube(int x)
            { side=x; }
};
int main()
{ Cube c1(10); Cube c2(20); Cube c3(30);
           cout << c1.side; cout << c2.side; cout << c3.side;
}
OUTPUT : 10 20 30
```

3. Copy Constructors:

The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. The copy constructor is used to:

Initialize one object from another of the same type.
Copy an object to pass it as an argument to a function.
Copy an object to return it from a function.

A copy constructor is a constructor of the form **classname(classname &).**The compiler will use the copy constructors whenever you initialize an instance using values of another instance of the same type.

Copying of objects is achieved by the use of a copy constructor and an assignment operator.

Note: ***The argument to a copy constructor is passed by reference, the reason being that when an argument is passed by value, a copy of it is constructed. But the copy constructor is creating a copy of the object for itself, thus, it calls itself. Again the called copy constructor requires another copy so again it is called. In fact it calls itself again and again until the compiler runs out of the memory .so, in the copy constructor, the argument must be passed by reference.***

DESTRUCTORS:

A destructor is also a member function whose name is the same as the class name but is preceded by tilde("~"). It is executed automatically by the compiler when an object is destroyed.

Destructors are usually used to deallocate memory and do other cleanup for a class object and its class members when the object is destroyed.

A destructor is called for a class object when that object passes out of scope or is explicitly deleted.

Example :

```
class TEST
{       int Regno,Max,Min,Score;
Public:
 TEST( )                                  // Default Constructor
 {                              }
                                          // Parameterized
 TEST (int Pregno,int Pscore)             Constructor
{
Regno = Pregno ;Max=100;Max=100;Min=40;Score=Pscore;
}
~ TEST ( )                                // Destructor
{ Cout<<"TEST Over"<<endl;}
};
```

Note:

Constructors and destructors do not have return type, not even void nor can they return values.

The compiler automatically calls constructors when defining class objects and calls destructors when class objects go out of scope.

Constructors can be overloaded.

Assignment No. 2

Q1. What are constructors and destructors? Give an example to illustrate the use of both.
Q2. Define Constructor overloading. Give an example.
Q3. Define Copy Constructor. Give an example.
Q4. Answer the questions (i) and (ii) after going through the following class :

```
class Test
    {
            char Paper[20];
            int Marks;
    public :
            Test()                          //Function 1
            {
            strcpy(Paper, "Computer");
            Marks = 0;
            }
            Test(char P[])                  //Function 2
            {
            strcpy(Paper, P);
            Marks = 0;
            }
            Test(int M)                     //Function 3
            {
            strcpy(Paper, "Computer");
            Marks = M;
            }
            Test(char P[],int M)            //Function 4
            {
            strcpy(Paper, P );
            Marks =M;
            }
    };
```

i. Which feature of Object Oriented Programming is demonstrated using Function 1, Function 2, Function 3, Function 4 in the above class Test?
ii. Write statements in C++ that would execute Function 2 and Function 4 of class Test.

Handout V: Inheritance

Inheritance is the process by which new classes called *derived* classes are created from existing classes called *base* classes.

The derived classes have all the features of the base class and the programmer can choose to add new features specific to the newly created derived class.

The idea of inheritance implements the **IS A** relationship. For example, mammal IS-A animal, dog IS-A mammal hence dog IS-A animal as well and so on.

Features or Advantages of Inheritance:

Reusability of Code
Saves Time and Effort
Faster development, easier maintenance and easy to extend
Capable of expressing the inheritance relationship and its transitive nature which ensures closeness with real world problems .

Base & Derived Classes:

A class can be derived from more than one classes, which means it can inherit data and functions from multiple base classes. A class derivation list names one or more base classes and has the form:

class derived-class: access-specifier base-class

Where access is one of **public, protected,** or **private**.

For example, if the *base* class is *MyClass* and the derived class is sample it is specified as:

class sample: public MyClass

The above makes sample have access to both *public* and *protected* variables of base class *MyClass*.

Consider a base class **Shape** and its derived class **Rectangle** as follows:

```
#include <iostream.h>

class Shape           // Base class
{
  public:
    void setWidth(int w)
    {             width = w;  }
    void setHeight(int h)
    {             height = h;  }
  protected:
    int width;  int height;
};
class Rectangle: public Shape // Derived class
{
  public:
    int getArea()
    {             return (width * height);  }
};

int main(void)
{
  Rectangle Rect;
  Rect.setWidth(5);
  Rect.setHeight(7);
```

```
  cout << "Total area: " << Rect.getArea() << endl;          // Print the area of the object.
  return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Total area: 35
```

Modes of Inheritance:

When deriving a class from a base class, the base class may be inherited through **public, protected** or **private** inheritance. The type of inheritance is specified by the access-specifier as explained above.

We hardly use **protected** or **private** inheritance, but **public** inheritance is commonly used. While using different type of inheritance, following rules are applied:

- **Public Inheritance:** When deriving a class from a **public** base class, **public** members of the base class become **public** members of the derived class and **protected** members of the base class become **protected** members of the derived class. A base class's **private** members are never accessible directly from a derived class, but can be accessed through calls to the **public** and **protected** members of the base class.
- **Protected Inheritance:** When deriving from a **protected** base class **public** and **protected** members of the base class become **protected** members of the derived class.
- **Private Inheritance:** When deriving from a **private** base class, **public** and **protected** members of the base class become **private** members of the derived class.

We can summarize the different access types according to who can access them in the following way:

Access	public	protected	private
Same class	Yes	yes	yes
Derived classes	Yes	yes	no
Outside classes	Yes	no	no

A derived class inherits all base class methods with the following exceptions:

Constructors, destructors and copy constructors of the base class. Overloaded operators of the base class.

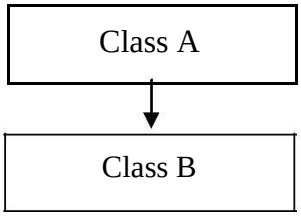
The friend functions of the base class.

Types of Inheritance:

1. Single class Inheritance:

Single inheritance is the one where you have a single base class and a single derived class.

Super/ Parent Class



Sub/Derived/Child class

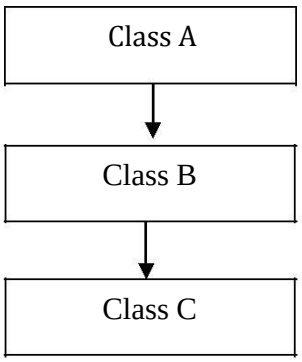
2. Multilevel Inheritance:

In Multi level inheritance, a subclass inherits from a class that itself inherits from another class.

A is a Base class (super) of B

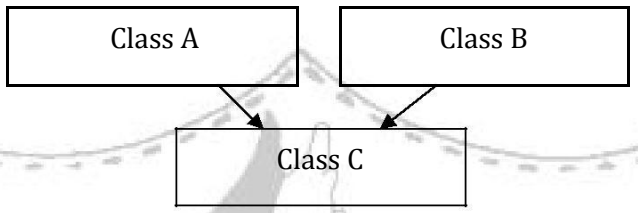
it is a sub class (derived) of A and base class of class C

sub class(derived) of class B



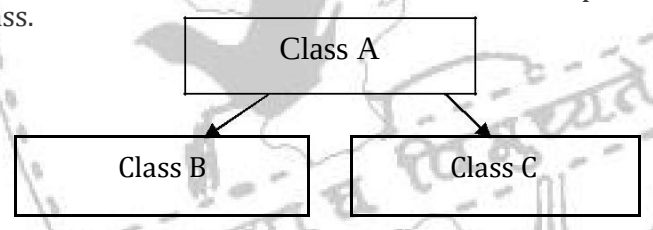
3. Multiple Inheritance:

In Multiple inheritances, a derived class inherits from multiple base classes. It has properties of both the base classes.



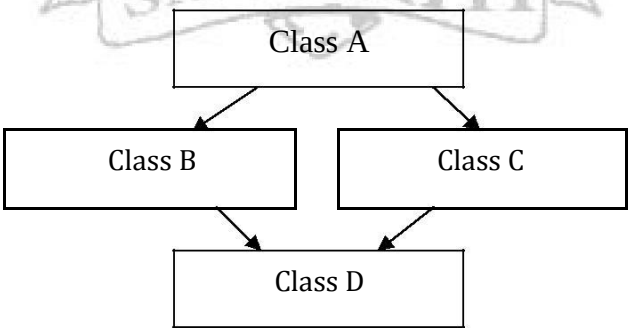
4. Hierarchical Inheritance:

In hierarchical Inheritance, it's like an inverted tree. So multiple classes inherit from a single base class.



5. Hybrid Inheritance:

a. It combines two or more forms of inheritance .In this type of inheritance, we can have mixture of number of inheritances but this can generate an error of using same name function from no of classes, which will bother the compiler to how to use the functions.
b. Therefore, it will generate errors in the program. This has known as ambiguity or duplicity.
c. Ambiguity problem can be solved by using **virtual base classes**



Multiple Inheritance Example:

A C++ class can inherit members from more than one class and here is the extended syntax:

```
class derived-class: access baseA, access baseB....
```

Where access is one of **public, protected,** or **private** and would be given for every base class and they will be separated by comma as shown above. Let us try the following example:

```
#include <iostream.h>
class Shape            // Base class Shape

{  public:
    void setWidth(int w)
    {              width = w;      }
    void setHeight(int h)
    {              height = h;     }
  protected:
    int width;
    int height;
};

class PaintCost        // Base class PaintCost
{  public:
    int getCost(int area)
    {              return area * 70;  }
};

class Rectangle: public Shape, public PaintCost // Derived class {
public:
    int getArea()
    {              return (width * height);  }
};
int main(void)
{ Rectangle Rect;
  int area;
  Rect.setWidth(5);
  Rect.setHeight(7);
  area = Rect.getArea();
  cout << "Total area: " << Rect.getArea() << endl; // Print the area of the object.
  cout << "Total paint cost: $" << Rect.getCost(area) << endl; // Print the total cost of painting
  return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Total area: 35
Total paint cost: $2450
```

Assignment No. 3

Q1. What is the need of inheritance in OOPS?

Q2. Differentiate between Abstract class and concrete class. Give one example of each. Q3. What do you understand by visibility modes?

Q4. Define the following :

a. Single Level Inheritance
b. Multi level Inheritance
c. Multiple Level Inheritance.

Give one example of each.

Q5. Consider the following and answer the questions that follow :

```
class School
{
    int A;
    protected : int B, int C;
    public :
                void INPUT ( int t);
                void OUTPUT ( );
};
class Dept : protected School
{
    int X,Y;
    protected :
                 void IN(int, int);
    public    :
                void OUT ( );
};
class Teacher : public Dept
{
    int P;
    void DISPLAY( );
    public :
                void ENTER( );
};
```

i) Name the Base class and Derived class of class Dept.
ii) Name the data members that can be accessed from function OUT( ).
iii) Name the private member function(s) of class Teacher.
iv) Is the member function OUT( ) accessible by the objects of Dept ?
v) How many bytes will be required by an object of class School and an object of class Teacher respectively.

Q6. Consider the following and answer the questions given below.

```
class University
{ int NOC;
   protected:
     char Uname[25];
   public:
```

```
        University() { }
        char State[25];
        void Enterdata();
        void Displaydata();
    };
class College : public University
    { int NOD;
      protected:
        void Affiliation();
      public:
        College()  {  }
        void  Enrol();
        void Show();
    };
class Department : public College
    {
        char Dname[25];
        int NOf;
      protected:
        void Affiliation();
      public:
        Department() { }
        void Input();
            void Display();
    };
```

i) Which class's constructor will be called first while declaring object of class Department?
ii) Which kind of inheritance is being depicted in the above example?
iii) If class college was derived privately from university, then, name members that could be directly accesses through the object(s) of class Department.
iv) How many bytes does an object belonging to the class Department require?
v) Which is the base class and the derived class of class College.

Q7. Identify the **syntax error**(s), if any (give reason for error) :

```
Class ABC
{  int  x  =  10;
     float y;
     ABC ( ) { y=5; }
     ~ABC ( ) { }
};
void main( )
{     ABC a1, a2;
}
```

Handout VI: Arrays

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.

Data Structure Arrays

Data structure array is defined as linear sequence of finite number of objects of same type with following set of operation:

- Creating : defining an array of required size
- Insertion: addition of a new data element in the in the array
- Deletion: removal of a data element from the array
- Searching: searching for the specified data from the array
- Traversing: processing all the data elements of the array
- Sorting : arranging data elements of the array in increasing or decreasing order
- Merging : combining elements of two similar types of arrays to form a new array of same type In C++ an array can be defined as

```
Datatype arrayname[size];
```

Where size defines the maximum number of elements can be hold in the array. For example

```
float b[10];//b is an array which can store maximum 10 float values
int c[5];
```

Array initialization

```
void main()
{
  int b[10]={3,5,7,8,9};
   cout<<b[4]<<endl;
 cout<<b[5]<<endl;
 }
```

Output is 9 0

In the above example the statement int b[10]={3,5,7,8,9} assigns first 5 elements with the given values and the rest elements are initialized with 0. Since in C++ index of an array starts from 0 to size-1 so the expression b[4] denotes the 5tn element of the array which is 9 and b[5] denotes 6tn element which is initialized with 0.

3	5	7	8	9	0	0	0	0	0
b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]

Searching

We can use two different search algorithms for searching a specific data from an array

- Linear search algorithm
- Binary search algorithm

Linear search algorithm

In Linear search, each element of the array is compared with the given item to be searched for. This method

continues until the searched item is found or the last item is compared.

```
#include<iostream.h>
int linear_search(int a[], int size, int item)
{
int i=0;

while(i<size&& a[i]!=item)
 i++;

if(i<size)
return i;//returns the index number of the item in the array else
return -1;//given item is not present in the array so it returns -1 since -1 is not a legal index number
}
void main()
{
int b[8]={2,4,5,7,8,9,12,15},size=8;
int item;
cout<<"enter a number to be searched
for"; cin>>item;
int p=linear_search(b, size, item); //search item in the array b
if(p==-1)
cout<<item<<" is not present in the array"<<endl;
else
cout<<item <<" is present in the array at index no "<<p;
}
```

In linear search algorithm, if the searched item is the first elements of the array then the loop terminates after the first comparison (best case), if the searched item is the last element of the array then the loop terminates after size time comparison (worst case) and if the searched item is middle element of the array then the loop terminates after size/2 time comparisons (average case). For large size array linear search not an efficient algorithm but it can be used for unsorted array also.

Binary search algorithm

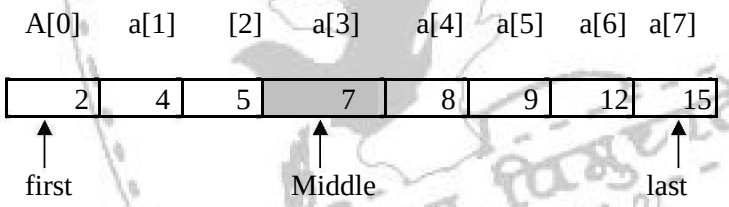
Binary search algorithm is applicable for already sorted array only. In this algorithm, to search for the given item from the sorted array (in ascending order), the item is compared with the middle element of the array. If the middle element is equal to the item then index of the middle element is returned, 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>
int binary_search(int a[ ], int size, int item)
{
int first=0,last=size-1,middle;
while(first<=last)
{
middle=(first+last)/2;
    if(item==a[middle])
            return middle; // item is found else if(item<
```

```
        a[middle])
                last=middle-1; //item is present in left side of the middle element
        else
                first=middle+1; // item is present in right side of the middle element
        }
return -1; //given item is not present in the array, here, -1 indicates unsuccessful search
}
void main()
{
int b[8]={2,4,5,7,8,9,12,15},size=8;
int item;
cout<<"enter a number to be searched for"; cin>>item;
int p=binary_search(b, size, item); //search item in the array b
if(p==-1)
        cout<<item<<" is not present in the array"<<endl;
else
        cout<<item <<" is present in the array at index no "<<p;
}
```

Let us see how this algorithm work for item=12 Initializing first =0 ; last=size-1; where size=8

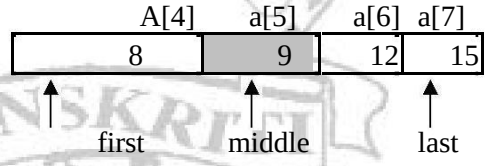
Iteration 1



First=0, last=7

middle=(first+last)/2=(0+7)/2=3 // note integer division 3.5 becomes 3 value of a[middle] i.e. a[3] is 7

7<12 then first= middle+1 i.e. 3 + 1 =4 **iteration 2**

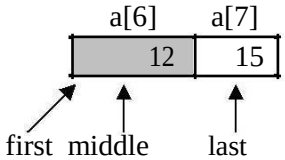


first=4, last=7 middle=(first+last)/2=(4+7)/2=5

value of a[middle] i.e. a[5] is 9 9<12 then

first=middle+1;5+1=6

iteration 3



first=6,last=7

Assignment No : 4
Arrays

Q 1. Define the term data structure?

Q 2. List out four important operations associated with linear data structure.

Q 3. What is the pre condition for Binary Search to be performed on a single dimensional array? Q 4. An array Val[1..15][1..10] is stored in memory with each element requiring 4 bytes of storage if the base address of array VAL is 1500, determine the location of VAL[12][9] when the array VAL is stored

i. Row Wise
ii. Column Wise

Q 5. An array S[40][30] is stored in the memory along the row with each of its element occupying 2 bytes, find out the memory location for the element S[15][5], if an element S[20][10] is stored to the memory location 5500.

Q 6. Write a function in C++ to combine the contents of two equi- sized arrays A and B by computing their corresponding elements with the formula 2*A[I]+3*B[I], where value I varies from 0 to N-1 and transfers the resultant content in the third same sized array.

Q 7. Write a user defined function in C++ to merge the contents of two sorted arrays A & B into third array C. Assume array A is sorted in ascending order, B is sorted in descending order, the resultant array is required to be in ascending order.

Q 8. Write a function which accepts an integer array and its size as arguments/ parameters and assign the elements into a two dimentional array of integers in the following format :

If the array is : 1, 2, 3, 4 , 5, 6

The resultant 2 D array is given below :
1 2 3 4 5 6 1 2 3 4 5 0 1 2 3 4 0 0 1 2 3
0 0 0 1 2 0 0 0 0 1 0 0 0 0 0

If the array is : 1, 2, 3

The resultant 2 D array is given below :
1 2 3 1 2 0 1 0 0

(**Hint** : *Use dynamic memory allocation to give memory to 2 D array*)

Assignment No : 5
Pointers

Q 1.What are Pointers? What is its use?
Q 2.What is the difference between :
- a) Arrays and Pointers
- b) Static and Dynamic memory allocation.
- c) **new** and **delete** operators.

Q 3.What is the use of following operators : *, &, →
Q 4.What do you understand by this pointer?
Q 5.What do you understand by Free Pool of memory?
Q 6.What are Memory Leaks. What are the possible reasons for it.

Q 7.Rewrite the following program after removing error(s), if any. Underline each correction.

a)
```
#  include<iostream.h>
struct triangle
{
        int a,b,c;
}
void input (triangle *T)
{
        T. a - =5;
        T. b /=2; T.
        c *= 0.5;
}
void display (triangle T)
{
        cout<<T. a << " : ";
        cout<<T. b << " : ";
        cout<< T.c<< " \n ";
}
void main ( )
{
        triangle A = 10, 9, 9 ;
        input(A);
        display(A);
        input(A);
        display(A);
        input(A);
        display(A);
}
```

b)
```
Void main( )
{ const int I; I =
        20;
        const int*ptr = &i ;
        (*ptr++);
        int j = 15;
        ptr = &j;
}
```

Q 8.Find the output of the following :

a)
```
# include <iostream.h>
#  include<conio.h>
void main ( )
{
    clrscr( );
    void func( int, int *); int
    a[5] = {2, 4, 6, 8, 10};
    int i, b = 5;
    for (i = 0; i<5; i ++)
    {
    func(a[i], &b);
    cout <<a[i] << "\t"<< b<< "\n";
    }
}
void func( int p, int *q);
{
    p = *q +=2;
}
```

b)
```
#  include<iostream.h>
void main ( )
{
        int array [ ] = {2, 3, 4, 5 };
        int *aptr=array;
        int value = *aptr ; cout<<value<< "\n";
        value = *aptr++ ; cout<<value<< "\n";
        value = *aptr ; cout<<value<< "\n";
        value = *++aptr ; cout<<value<< "\n";
}
```

c)
```
#  include<iostream.h>
void main( )
{
        int arr [ ] = {4, 6, 10,
        12}; int * ptr = arr;
        for (int i = 1; i<= 3; i++)
        {
                cout<<*ptr << "#";
                ptr++;
        }
        for (i = 1; i<= 2; i++)
        {
                (*ptr)* = 3;
                cout<<*ptr;
                 - -ptr ;
        }
        for (i = 0; i< 4; i++)
                cout<<arr[i] << "@";
        cout<<endl;
```

```
}
```

d)
```
void main()
{
    int x [] = {10, 20, 30, 40, 50};
    int *p, **q, *t;
    p = x; t
    = x+1;
    q = &t;
    cout<<*p<< "," <<**q<< "," << *t++;
}
```

e)
```
# include<iostream.h>
# include<string.h>
class state
{
    char *s_name;
    int size;
public:
    state( ){size =0;s_name = new car [size+1]; }
    state( char *s)
    {
        size = strlen(s);
        s_name = new car [size+1];
        strcpy(s_name,s);
    }
    void display( )
    {
        cout<<s_name<<endl;
    }
    void replace(state & a, state &b)
    {
        size = a.size +b.size;
        delete s_name;
        s_name = new car [size+1];
        strcpy(s_name,  a.s_name);
        strcat(s_name, b.s_name);
    }
};
void main()
{
    char * temp = "Delhi";
    state state1(temp), state2("Mumbai"),
    state3("Nagpur"),s1,s2; s1.replace(state1, state2);
    s1.replace(s1, state3);
    s1.display( );
    s2.display( );
}
```

Q 9.Point out the errors in the following C++ statements :

a) const int ***pc ; *pc =10; (***pc)++;

b) float num = 3.10, const *pc = #
float num1= 5.8; pc =&num1;

c) float x = 12.5;
float xptr =&x;

d) float *rptr;
int *iptr;
iptr = fptr;

Assignment No. 6
File Handling

Q 1.Which is the base class of all the classes? Q
2.Which are the two ways of opening a file?
Q 3. Why is it not essential to include iostream.h if fstream.h is included in the program?
Q 4.Differentiate between:

a) ios:: ate and ios app
b) Binary Files and Text files
c) cin.getline( ) and cin.get( )
d) ios:: nocreate and ios:: noreplace
e) tellg( ) and tellp( )
f) seekg( ) and seekp( )

Q 5.Assuming the class STOCK, write user defined functions in C++ to perform the following :

a) Write the objects of STOCK to a binary file
b) Read the objects of STOCK from binary file and display all the records with price less than 200 on the screen.

```
class STOCK
{
        int item_no;
        char item[10];
        float price;
public :
        void entry( )
        {
                cin>>item_no;
                gets(item);
                cin>>price;
        }
        void show( )
        {
                cout<< item_no << item << price;
        }
};
```

Q 6. Assuming the class employee given below, complete the definitions of the member functions.

```
class employee
{
        int emp_no;
        char name[10];
        float salary;
public :
        void entry( );          // Input data from the user.
        void show( );           // Display data on the screen.
        void write_file( )      // Write data to the file from an object.
        void read_file( )       // Read data from the file to an object.
};
```

Q 7. Given the binary file sports.dat, containing records of the following structure type:

```
Struct Sports
{
        int code;
        char Event[20];
        char Participant[10][30];
        int no_of_participants;
};
```

a) Write a function in C++ that would read contents from the file sports.dat and create a file named athletic.dat copying only those records from sports.dat where the event name is "Athletics".

b) Write a function in C++ to update the file with the new value of no_of_participants. The value of code and no_of_participants are read during the execution of the program.

Q 8. Write a function in C++ to print the count of the world **the** as anindependent word in the text file story.txt.
For Example , if the content of the file story.txt is:
There was a monkey in <u>the</u> zoo. <u>The</u> monkey was very naughty.
Then the output of the program should be 2.

Q 9. Consider the code given below :

```
void main( )
{
    char ch = 'A';
    fstream fout ("data.dat",
    ios::out); fout<<ch;
    int p = fout.tellg( );
    cout<<p;
}
```

What is the output if the file content before the execution of the program is the string "ABC".

Q 10. Observe the program segment given below carefully and fill the blanks marked as statement 1 and statement 2 using seekg( ) and seekp( )functions for performing the required task.

```
# include <fstream.h>
class Item
{
        int Ino;
        char item[20];
        public :

// Function to search and display the content from a particular record number
        void search (int);
// Function to modify the content of a particular record number
        void Modify (int);
```

```
};
void item :: Search( int RecNo)
{
        fstream file;
        file.open ("stock.dat", ios::binary|ios::in);
        _____ //Statement 1
        file.read((char*)this, sizeof(item));
        cout<<Iino<< " "<<Item;
        file.close ( );
}
void item :: Modify( int RecNo)
{
        fstream file;
        file.open ("stock.dat",
        ios::binary|ios::in|ios::out); cout>>Ino;
        cin.getline(item,20);
        _____ //Statement 2
        file.write((char*)this, sizeof(item));
        cout<<ino<< " "<<Item;
        file.close ( );
}
};
```

Q 11. Following is the structure of each record in the data file named "Product.dat".

```
struct Product
{
        char P_code[10];
        char P_Description [10];
        int stock;
};
```

Write a function in C++ to update the file with a new value of Stock. The stock and the P_code, whose stock is to be updated are read during the execution of the program.

Q 12. Following is the structure of each record in the data file named "Flights.dat".

```
struct Flight
{
        int Fl_No;
        char Starting [20];
        char Ending [20];
};
```

Write a function in C++ to delete a record from the file. The Fl_No, whose record is to be deleted is read during the execution of the program.

Assignment No : 7
Linked list, Stacks and Queues

Q 9.Define Linked List? What is its advantage over arrays.

Q 10. What is AVAIL list.

Q 11. Differentiate between circular queues and d-queues.

Q 12. Each node of a **STACK** contains the following information, in addition to the required pointer field :

a. acc_no
b. acc_name

Give the structure of node for the linked stack in question

TOP is a pointer which points to the topmost node of the **STACK**. Write the following functions.

i) **PUSH( )** – *To push a node to the **STACK** which is allocated dynamically.*

ii) **POP ( )** – *To remove a node from the **STACK** and release the memory.*

Q 13. Change the following infix expression to its postfix equivalent:

A +B * (- C) / A ^D

Show stack after each pass.

Q 14. Solve the following **postfix** expression using the stack method, for the given values.

A B + C * D E / +

A=3, B=5, C=2, D=8, E=4

Show stack after each pass.

Q 15. Solve the following postfix expression using a stack. TRUE FALSE and FALSE TRUE not or and

Show the contents of stack after execution of each operation:

Q 16. Given the following class :

```
Char *msg[ ] = { "overflow", " underflow" };
class stack
{
        int top;        //the stack pointer
        int stk [5];    // the elements
        void err_rep( int e_enum) { cout<< msg[e_enum]; }
public :
        stack ( ) { top=-1;} //function to initialize the stack pointer
        void push(int);         // put new value in stk
        void pop ( );           // get the top value and display it
};
```

Define the member functions push( ) and pop( ) outside the class, the functions should invoke err_rep( ) in case of overflow/ underflow.

Q 17. Given the following class :

```
Char *msg[ ] = { "overflow", " underflow" };
class queue
```

```
 {
        int front, rear;        //the queue pointers
       int que [15];  // the elements
       void err_rep( int e_enum) { cout<< msg[e_enum]; }
public :
       queue ( ) { front=-1;rear=-1} //function to initialize the queue pointers
       void inst(int);              // add a new value in queue
       void remove ( );             // remove an element from the queue &
                                    // display it
};
```

Define the member functions inst( ) and remove( ) outside the class, the functions should invoke err_rep( ) in case of overflow/ underflow.

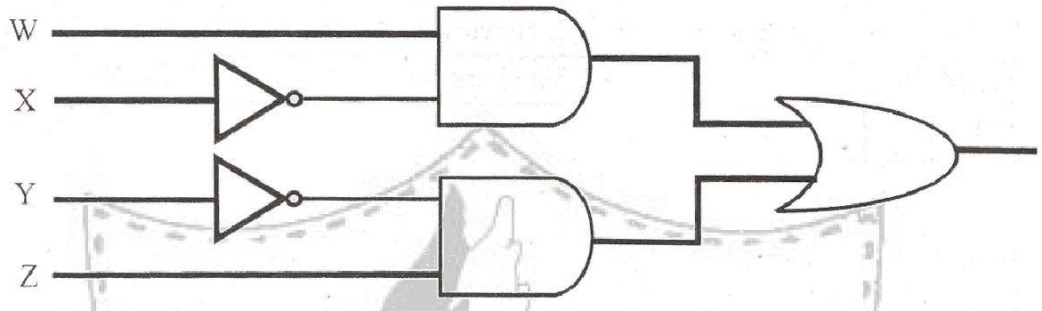
Q 18. Write functions to perform insert and delete operations on a circular queue of integers. (Array Implementation)

Assignment No : 8
Boolean Algebra

1. (a) State Distributive law and verify the same using truth table.
   (b)Write the equivalent Canonical Sum of Product expression for the following Product of Sum Expression

   F(X,Y,Z)=∏(1, 3, 6, 7)

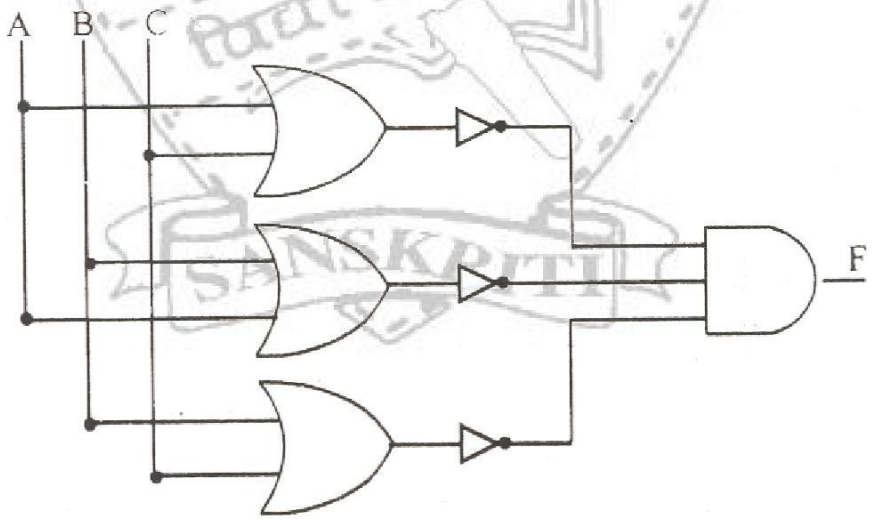
   (c)Write the equivalent Boolean Expression for the following Logic Circuit.



   (d)Reduce the following Boolean expression using K-Map
   F(U,V,W,Z) = Σ(0, 1, 2, 3, 4, 10, 11)

2. (a) State and verify Associative Law

   (b)Write the equivalent expression for the following logical circuit



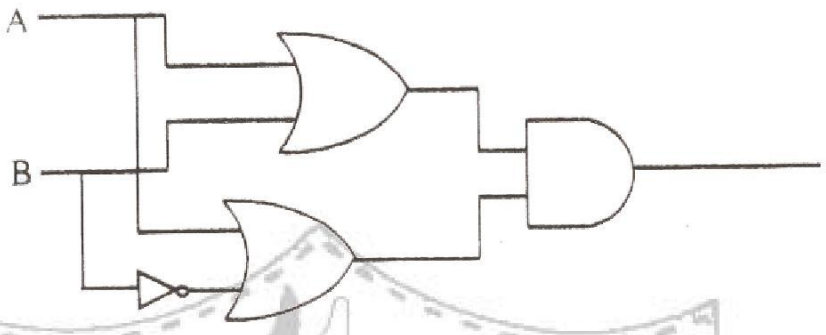
   (c) Express **P + Q'R** in canonical **POS** form.

   (d)Reduce the following Boolean expression using K-map

   F(P,Q,R,S) = ∏ (0,3, 5, 6, 7, 11, 12, 15)

3. (a) State and verify DeMorgan's Law algebraically

(b) Write the equivalent Boolean expression for the following logic circuit:



(c) Represent the Boolean expression **x.y' + y.z'** with the help of NAND gates only.

(d) Obtain simplified form of the following Boolean expression using Karnaugh Map

F(x,y,z,w) = Σ(1, 3, 4, 5, 7, 9, 11, 12, 13, 15)

(e) Write the POS form of Boolean Function F(U,V,W) which is represented in the truth table as :

U	V	W	F
0	0	0	0
0	0	1	0
0	I	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Assignment No. 9
Communication And Network Concepts

Q 1.Name two transmission media for networking.

Q 2.Give the full form of the following :

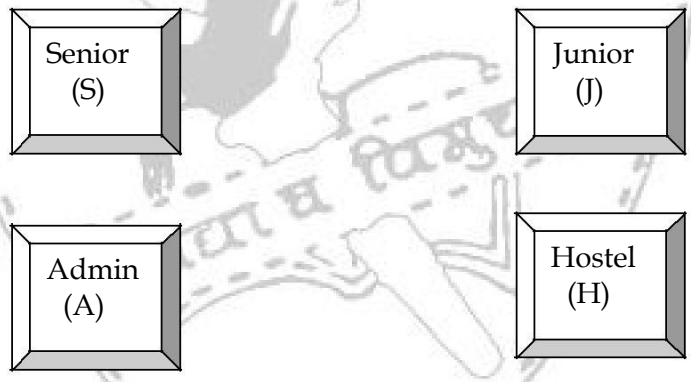
i. GSM
ii. XML
iii. FTP
iv. XML

Q 3.Differentiate between Hackers and Crackers.

Q 4.What is the difference between LAN and WAN?

Q 5.What is the purpose of having web browser? Name any one commonly used browser.
Q 6.Give two major reasons to have network security.

Q 7.Indian Public School in Darjeeling is setting up the network between its different wings. It has four wings named as Senior (S), Junior(J), Admin(A) and Hostel(H):



Distance between various Wings :

Wing A to Wing S	100 m
Wing A to Wing J	200 m
Wing A to Wing H	400 m
Wing S to Wing J	300 m
Wing S to Wing H	100 m
Wing J to Wing H	450 m

Number of Computers :

Wing A	10
Wing S	200
Wing J	100
Wing H	50

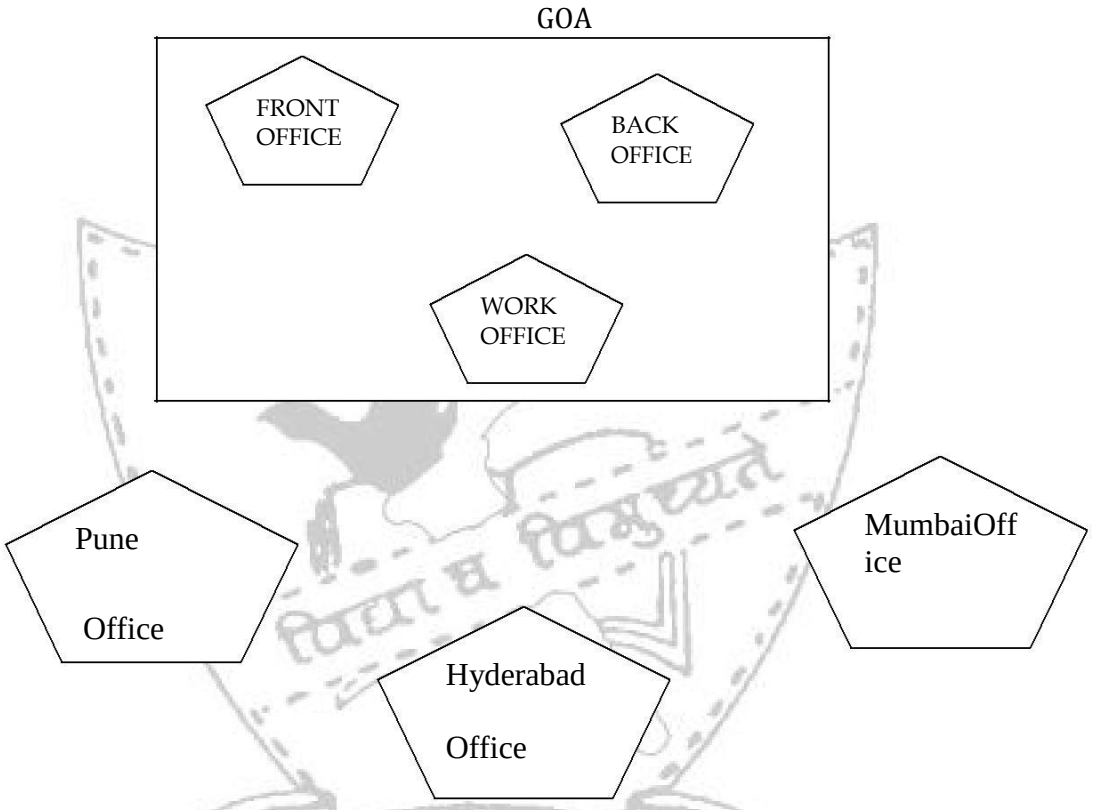
a. Suggest a suitable Topology for networking the computer of all the wings.
b. Suggest the most suitable wing to house the server of this school. Justify your answer.
c. Suggest the placement of the following devices with justification :

i. Repeater
ii. Hub/ Switch

d. Mention an economic technology to provide internet accessibility to all the wings.

Q 8."United Group" is planning to start their offices in four cities of India.. The company has planned to set up their head office in Goa in three locations and have named their Goa offices as "Front Office", "Back Office" and "Work office". The company's regional offices are located at "Pune", "Mumbai" and "Hydrabad".

A rough layout of the same is as follows:



Distance between various Wings :

Place From	Place to	Distance
Front Office	Back Office	15 K.M.
Front Office	WorkOffice	35 K.M.
Front Office	Pune Office	350K.M.
Front Office	Bombay Office	300 K.M.
Front Office	Hydrabad Office	1000 K.M.

Number of Computers :

Front Office	150
Back Office	70
Work Office	20
Pune Office	65
Mumbai Office	90
Hyderabad Office	100

a. Suggest a network type(out of LAN, MAN, WAN) for connecting each of the following set of their offices :
   - Front Office and Back Office
   - Front Office and Pune Office

b. Which device will you suggest to be procured by the company for connecting all the computers within each of their offices out of the following devices?
   - Modem
   - Telephone
   - Switch/ Hub

c. Which of the following communication media, will you suggest to be procured by the company for connecting their local offices in Goa for very effective and fast communication ?
   - Ethernet Cable
   - Optical Fiber
   - Telephone Cable

d. Suggest a cable/ Wiring layout for connecting the company's local offices located in Goa. Also, suggest an effective method/ technology for connecting the company's regional office at "Pune", "Bombay" and "Hydrabad".

Assignment No. 10.1
Structured Query Language

Q 1.What is a database system? What is its need?

Q 2.What is a relation?

Q 3.What is a Primary Key?

Q 4.What is an Alternate Key?

Q 5.What is a candidate Key?

Q 6.What do you understand by degree of relation?

Q 7.What do you understand by cardinality of relation?

Q 8.Define the following :

i. Attribute
ii. Tuple
iii. Foreign Key
iv. Data redundancy
v. DDL
vi. DML

Structured Query Language
Assignment No : 10.2

Write SQL commands for the following :

Q. 1 Create a table with the following specifications :

TABLE : BANK

Field name	Type	Description
Acc_no	Int	PRIMARY KEY
CName	char(20)	NOT NULL
Address	Char(30)	NOT NULL
Ph_No	Int	Can have NULL values
Amount	FLOAT	Minimum 5000
DOO	Date	Date of opening the account ; NOT NULL
Type	Char(1)	Type of account (S->savings, C->current, R->recurring)

Q.2 Insert the following records in the above table (*write commands for any three*):

Acc no	CName	Address	Ph No	Amount	DOO	Type
1	Karan	22/7, green Park, Delhi	321345	45000	12/01/1975	S
2	Puneet	3/1, Friends colony, Delhi	324567	89000	01/01/1985	R
3	Anirban	12/7, GK-1, New Delhi	432664	78000	11/10/1988	C
4	Yatin	11, A Block, Rohini, Delhi	256997	100000	06/08/1999	S
5	Sonia	32/4, green Park, Delhi	123456	8921	03/08/1998	S
6	Sophia	13/11, Friends colony, Delhi	NULL	45697	05/08/1996	R
7	Nikhil	112/7, GK-1, New Delhi	234890	14752	12/01/1975	C
8	Tarun	21, A Block, Rohini, Delhi	NULL	5500	01/01/1986	R
9	Jisha	113, A Block, Rohini, Delhi	126790	78451	11/10/1978	R
10	Tanmay	1/7, GK-1, New Delhi	345678	70000	06/08/1989	S

Considering the above table write SQL commands for the following

: Q 3. Display the record of all the customers.

Q 4. Display the record of all the customers sorted in descending order of their amounts.

Q 5. To count total number of customers for each type.

Q 6. To display the total money deposited in the bank.

Q 7. To list out the names of all the customers, whose name starts with **S**.

Q 8. To display the Acc_no and names of all those customers, who do not have phone.

Q 9. To display total amount in each type of account.

Q 10. To list the names and addresses of all the customers staying in Rohini.

Structured Query Language
Assignment No. 10.3

Write SQL commands for the following :

Q. 1 Create a table with the following specifications :

TABLE : SUPPLIER

Field name	Type	Description
S_no	Char(2)	PRIMARY KEY
PName	char(20)	NOT NULL
SName	Char(20)	NOT NULL
Qty	Int	Should be in the range of 100 to 500
Price	FLOAT	Should not be more than 50
City	Char(15)	Should be any of the four metropolitan cities.

Q.2 Insert the following 10 records in the above table (Write commands for any three):

S No	PName	SName	Qty	Price	City
S1	Bread	Britania	150	8.00	Delhi
S2	Cake	Britania	250	20.00	Mumbai
S3	Coffee	Bru	170	45.00	Delhi
S4	Chocolate	Amul	380	10.00	Kolkata
S5	Souce	Kissan	470	36.00	Chennai
S6	Maggi	Nestle	340	10.00	Mumbai
S7	Biscuit	Britania	560	21.00	Delhi
S8	Jam	Kissan	220	40.00	Kolkata
S9	Tea	TATA	345	45.00	Chennai

Considering the above table answer the questions that follow :

Q 3. Display the record of all the suppliers.

Q 4. Display data of all the products whose quantity is between 170 and 370.

Q 5. Give Sname for the products whose name starts with 'C'.

Q 6. Sname, Pname,Price for all the products whose quantity is less than 200.

Q 7. To list out the names of all the products, whose name ends with **S**.
Q 8. To display S_No, Pname, Sname, Qty in descending order of qty.

Q 9. To count the number of distinct cities.

Q 10. To list the Snames, Qty, Price and value (Qty*Price) of each product.

Structured Query Language
Assignment No : 10.4

Write SQL commands for the following :

Q. 1 Create a table with the following specifications :

TABLE : CLUB

Field name	Type	Description
Coach_id	int	PRIMARY KEY
CName	char(20)	NOT NULL
Age	int	Should be between 28 to 40
Sports	Char(20)	Should be either of Karate, Squash, Basket Ball, Swimming
Pay	FLOAT	Should be between 1000 to 2500
DOA	Date	Date of Appointment
Sex	Char(1)	M- male, F- Female

Q.2 Insert the following 10 records in the above table (Write commands for any three) :

Coach id	CName	Age	Sports	Pay	DOA	Sex
1	Karan	35	Karate	1000	12/01/1995	M
2	Puneet	34	Karate	1200	01/01/1995	M
3	Anirban	34	Squash	2000	11/10/1998	M
4	Yatin	33	Basket Ball	1500	06/08/1999	M
5	Sonia	36	Swimming	1100	03/08/1998	F
6	Sophia	36	Swimming	2200	05/08/1996	F
7	Nikhil	39	Squash	2400	12/01/1995	M
8	Tarun	37	Karate	2500	01/01/1996	M
9	Jisha	41	Swimming	2300	11/10/1998	F
10	Tanmay	37	Basket Ball	2200	06/08/1999	M

Considering the above table answer the questions that follow :

Q 3. To display all the information about the swimming coaches in the club.

Q 4. Display the names of all the coaches with their date of appointment in descending order.

Q 5. To display a report, showing coach name, pay, age and bonus(15% of pay) for all the coaches.

Q 6. To display the total number of coaches in the club.

Q 7. To count the total number of male and female coaches.

Q 8. To display total salary given to all the coaches.

Q 9. To display the salaries of all the coaches appointed after {01/01/1998}.

Q 10. To increment the Pay of all the coaches by 500, earning less than 1500.

Structured Query Language
Assignment No : 10.5

Write SQL commands for the following :

Q. 1 Create a table with the following specifications :

TABLE : VOTER

Field name	Type	Description
V_no	int	PRIMARY KEY
VName	char(20)	NOT NULL
Address	Char(30)	NOT NULL
Ph_No	Int	Can have NULL values
Age	int	Should be >=18

Q.2 Insert the following 10 records in the above table (Write commands for any three) :

V_no	VName	Address	Ph_No	Age
1	Kiran	22/7, C R Park, Delhi	321345	32
2	Puneeta	113/1, Friends colony, Delhi	324567	18
3	Anubhav	12/7, GK-1, New Delhi	432664	47
4	Yatin	11, A Block, Rohini, Delhi	256997	22
5	Suniana	32/4, green Park, Delhi	123456	32
6	Sophia	13/11, Friends colony, Delhi	NULL	89
7	Nakul	112/7, GK-1, New Delhi	234890	41
8	Tarang	21, A Block, Rohini, Delhi	NULL	85
9	Nisha	113, A Block, Rohini, Delhi	126790	78
10	Tanmay	1/7, GK-1, New Delhi	345678	65

Considering the above table answer the questions that follow :

Q 3. Display the record of all the voters.

Q 4. Display the V_No, Vname, age of all the voters sorted in descending order of their names.

Q 5. To count total number of voters, staying in Friends colony.

Q 6. To change the phone number of the person whose voter number are 5 to 326768.

Q 7. To list out the names of all the voters who are more than 75 years of age.

Q 8. To display name and address of all the voters who are of more than 60 years of age.

Q 9. To count, total number of voters.

Q 10. To list the names and addresses of all the voters whose name start with **T** and end with **g.**

Structured Query Language
Assignment No : 10.6

Write SQL commands for the following :

Q. 1 Create a table with the following specifications :

TABLE : STUDENT

Field name	Type	Description
No	int	PRIMARY KEY
Name	char(20)	NOT NULL
Stipend	Float	Value should range bet. 400 & 600
Stream	Char(15)	Medical, Commerce, Humanities, Non Medical
AvgMarks	Float	Value should range bet. 0 & 100
Grade	Char(1)	Marks Grade >=75 A 50- 74 B <=49 C

Q.2 Insert the following 10 records in the above table (write commands for any three) :

No	Name	Stipend	Stream	AvgMark	Grade
1	Diwakar	400.00	Medical	78.5	A
2	Deepa	550.00	Commerce	89.5	A
3	Divya	600.00	Non Medical	68.6	B
4	Arun	475.00	Commerce	49.0	C
5	Sabina	425.00	Medical	40.25	C
6	John	600.00	Humanities	94.5	B
7	Robert	500.00	Commerce	88.5	A
8	Rubina	550.00	Non Medical	92.5	A
9	Vikas	575.00	Medical	67.5	B
10	Mohan	600.00	Humanities	73.0	A

Considering the above table answer the questions that follow :

Q 3. Display the records of all non medical students.

Q 4. List all the students sorted by their Average marks.

Q 5. To display a report, listing name, stipend, stream and amount of stipend received in a year assuming stipend is paid every month.

Q 6. To count the number of students with grade 'A'.

Q 7. To display average of average marks scored by the students of each stream.

Q 8. To count the number of medical students who are getting stipend greater than 500.

Q 9. To display total amount of stipend given to all the students per annum.

Q 10. To count the number of students with grade **A**.

Program List

General Instructions:

1. Practical files should contain program listings and outputs.
2. Each program should be headed with **Program number**.
3. Header comment of each program should contain :
   Name of the Program : _____
   Programmers Name :_____
   Date : _____

Program#	PROGRAMS	Date of Submission
1	Create a header file date.h to define a structure **date** with the following data members : dd int mm int yy int Now write the following functions : a)To input a date b)To display the date c)To verify the validity of the date.(return the status)	
2	Write a menu driven program to execute the following functions : ▪ Reads a decimal number and returns its octal equivalent. ▪ Reads an octal number and returns its decimal equivalent.	
3	Define a structure **student** with the following members : Rollno **int** Name **char** [20] Marks **float**[5] Total **float** Per **float** DOB **date** (use date.h) Now, Write the following functions : 1. *To **input** rollno, name, marks in 5 subjects, date (use input and validate( ) from **date.h**) and calculate total & percentage of a student* 2. *To **display** the data of the students.* Write a menu driven function to **execute** the above functions for 5 students.	
4	Define a class **Account** with the following **Data members** :	

	Account Number **int** (Automatically Generated) Customer Name **char** [20] Type of Account **char (S ->** Saving Account **C** -> Current Account) Balance **float (**Minimum balance -> 1000) **Member Functions** : *1) Constructor to initialise the data members.* *2) To input the data members.* *3) To withdraw money after checking the balance.* *4) To display the data members.* Write a menu driven program to execute the above class for 5 account holders.	
5	WAP using classes to Add, Subtract and Multiply two 2 – D Arrays (Matrices).	
6	WAP using classes to implement the following on a 2-D integer array (Matrix). 1. Sum of Rows. 2. Sum of columns 3. Sum of all the elements	
7	WAP using classes to implement the following on a 2-D integer array (Matrix). 1. Sum of Left diagonal 2. Sum of Right diagonal 3. Print upper half of the diagonal 4. Print lower half of diagonal	
8	Define a class **Product** with the following **Data members** : Product Number **int** (Automatically Generated) Product Name **char** [20] Price float **Member Functions** : *1) To input the data members.* *2) To display the data members.* **Global Functions:** *1) A function to sort the records (using Insertion Sort)of all products in the ascending order of their price.* *2) To insert a new record in the sorted array without altering the order of the records.* *3) To search for a Product(Using Binary Search)in the array of products.* Write a menu driven program to execute the above class	

	and functions for 5 Products	
9	WAP to copy the contents of one text file to another	
10	WAP to count the number of words in a text file.	
11	Define a class **Tour** with the following **Data members** : TCode int NoofAdults int NoofKids int Kilometers int TotalFare float **Member Functions** : *1) To input the data members.* *2) To display the data members.* *3) To and write the data to a Binary File* *4) To read the data from the Binary file* Write a menu driven program to execute the above class.	
12	WAP to swap two strings from an array of strings using pointers.	
13	WAP using classes/ structures to : 1. Create a Link List. 2. Display the List 3. Insert a Node in it. 4. Delete a Node from it.	
14	WAP using classes/ structures to : (USING ARRAYS) 1. Create a Stack 2. Push an element into it. 3. Pop an element out of it.	
15	WAP using classes/ structures to : (USING LINK LIST) 1. Create a Stack 2. Push an element into it. 3. Pop an element out of it.	
16	WAP using classes/ structures to : (USING ARRAYS) 1. Create a Queue 2. Insert an element into it. 3. Delete an element from it.	

17	WAP using classes/ structures to : (USING LINK LIST) 1. Create a Queue 2. Insert an element into it. 3. Delete an element from it.	

References for Study Material and Notes

www.mycbseguide.com
www.tutorialspoint.com
www.cplusplus.com/