

**OBJECT ORIENTED PROGRAMMING  
LABORATORY MANUAL**



**Computer Science and  
Engineering**

## CONTENTS

Week	Name of the program	P no																						
1	Study of C++ Standard library functions	1																						
2	a) Write a C++ program to find the sum of individual digits of a positive integer. b) Write a C++ program to generate the first n terms of the sequence.	3																						
3	a) Write a C++ program to generate all the prime numbers between 1 and n, where n is a value supplied by the user. b) Write a C++ program to find both the largest and smallest number in a list of integers.	5																						
4	a) Write a C++ program to sort a list of numbers in ascending order. b) Write a Program to illustrate New and Delete Keywords for dynamic memory allocation	7																						
5	a) Write a program Illustrating Class Declarations, Definition, and Accessing Class Members. b) Program to illustrate default constructor, parameterized constructor and copy constructors c) Write a Program to Implement a Class STUDENT having Following Members: <table border="1" style="margin-left: 40px; margin-right: 40px;"> <thead> <tr> <th style="text-align: center;">Member</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td colspan="2" style="text-align: center;"><b>Data members</b></td> </tr> <tr> <td>sname</td> <td>Name of the student</td> </tr> <tr> <td>Marks array</td> <td>Marks of the student</td> </tr> <tr> <td>total</td> <td>Total marks obtained</td> </tr> <tr> <td>tmax</td> <td>Total maximum marks</td> </tr> </tbody> </table> <table border="1" style="margin-left: 40px; margin-right: 40px;"> <thead> <tr> <th colspan="2" style="text-align: center;">Member functions</th> </tr> <tr> <th style="text-align: center;">Member</th> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>assign()</td> <td>Assign Initial Values</td> </tr> <tr> <td>compute()</td> <td>to Compute Total, Average</td> </tr> <tr> <td>display()</td> <td>to Display the Data.</td> </tr> </tbody> </table>	Member	Description	<b>Data members</b>		sname	Name of the student	Marks array	Marks of the student	total	Total marks obtained	tmax	Total maximum marks	Member functions		Member	Description	assign()	Assign Initial Values	compute()	to Compute Total, Average	display()	to Display the Data.	9
Member	Description																							
<b>Data members</b>																								
sname	Name of the student																							
Marks array	Marks of the student																							
total	Total marks obtained																							
tmax	Total maximum marks																							
Member functions																								
Member	Description																							
assign()	Assign Initial Values																							
compute()	to Compute Total, Average																							
display()	to Display the Data.																							
6	a) Write a Program to Demonstrate the i) Operator Overloading. ii) Function Overloading. b) Write a Program to Demonstrate Friend Function and Friend Class.	16																						
7	a) Write a Program to Access Members of a STUDENT Class Using Pointer to Object Members. b). Write a Program to Generate Fibonacci Series use Constructor to Initialize the Data Members.	20																						
8	<b>Revision of Programs</b>																							
9	Write a C++ program to implement the matrix ADT using a class. The operations supported by this ADT are: a) Reading a matrix. b) Addition of matrices. c) Printing a matrix. d) Subtraction of matrices. e) Multiplication of matrices	22																						

<b>10</b>	Write C++ programs that illustrate how the following forms of inheritance are supported: a)Single inheritance c)Multi level inheritance b)Multiple inheritance d)Hierarchical inheritance	<b>27</b>
<b>11</b>	a.)Write a C++ program that illustrates the order of execution of constructors and destructors when new class is derived from more than one base class. b) Write a Program to Invoking Derived Class Member Through Base Class Pointer.	<b>34</b>
<b>12</b>	a) Write a Template Based Program to Sort the Given List of Elements. b) Write a C++ program that uses function templates to find the largest and smallest number in a list of integers and to sort a list of numbers in ascending order.	<b>38</b>
<b>13</b>	a) Write a Program Containing a Possible Exception. Use a Try Block to Throw it and a Catch Block to Handle it Properly. b) Write a Program to Demonstrate the Catching of All Exceptions.	<b>41</b>
<b>14</b>	<b>Revision of programs</b>	

## **Week-1**

### **C++ Standard Library**

The C++ Standard Library can be categorized into two parts:

- **The Standard Function Library:** This library consists of general-purpose, stand-alone functions that are not part of any class. The function library is inherited from C.
- **The Object Oriented Class Library:** This is a collection of classes and associated functions.

Standard C++ Library incorporates all the Standard C libraries also, with small additions and changes to support type safety.

#### **The Standard Function Library:**

The standard function library is divided into the following categories:

- I/O
- String and character handling
- Mathematical
- Time, date, and localization
- Dynamic allocation
- Miscellaneous
- Wide-character functions

#### **The Object Oriented Class Library:**

Standard C++ Object Oriented Library defines an extensive set of classes that provide support for a number of common activities, including I/O, strings, and numeric processing. This library includes the following:

- The Standard C++ I/O Classes
- The String Class
- The Numeric Classes
- The STL Container Classes
- The STL Algorithms

- The STL Function Objects
- The STL Iterators
- The STL Allocators
- The Localization library
- Exception Handling Classes
- Miscellaneous Support Library

## Week-2

**2.a) Write a C++ program to find the sum of individual digits of a positive integer.**

### Program:

```
#include<iostream.h>
intsum_of_digits(int n)
{
    intdigit,sum=0;
while(n!=0)
    {
        digit=n%10;
        sum=sum+digit;
        n=n/10;
    }
return sum;
}
int main()
{
    intnumber,digits_sum;
    cout<<"Enter Positive integer within the range:";
    cin>>number;
    digits_sum=sum_of_digits(number);
    cout<<"sum of digts of "<<number<<" is "<<digits_sum;
    return 0;
}
```

### Input:

Enter Positive integer within the range:4321

### Output:

sum of digits of 4321 is 10

---

**2.b) Write a C++ Program to generate first n terms of Fibonacci sequence.**

**Program:**

```
#include<iostream.h>
void fib(int n)
{
    int f0,f1,f,count=0;
    f0=0;
    f1=1;
    while(count<n)
    {
        cout<<f0<<"\t";
        count++;
        f=f0+f1;
        f0=f1;
        f1=f;
    }
}
int main()
{
    int terms;
    cout<<"Enter How many terms to be printed:";
    cin>>terms;
    fib(terms);
    return 0;
}
```

**Input:**

Enter How many terms to be printed:10

**Output:**

0 1 1 2 3 5 8 13 21 34

---

**Week-3**

**Write a C++ program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.**

**Program:**

```
#include<iostream.h>
void prime(int n)
{
    int factors;
    cout<<"prime numbers are... ";
    for(int i=1;i<=n;i++)
    {
        factors=0;
        for(int j=1;j<=i;j++)
        {
            if(i%j==0)
                factors=factors+1;
        }
        if(factors<=2)
            cout<<i<<"\t";
    }
}
int main()
{
    int n;
    cout<<"Enter a integer value:";
    cin>>n;
    prime(n);
    return 0;
}
```

**Input:**

Enter a integer value:10

**Output:**

prime numbers are....1 2 3 5 7

**Write a C++ Program to find both the largest and smallest number in a list of integers.**

**Program:**

```
#include<iostream.h>
int main()
{
    int a[50],i,n,small,large;
    cout<<"Enter The Array Size:";
    cin>>n;
    cout<<"ENTER ELEMENTS OF ARRAY";
    for(i=0;i<n;i++)
    cin>>a[i];
    small=a[0];
    large=a[0];
    for(i=0;i<n;i++)
    {
        if(a[i]<small)
            small=a[i];
        if(a[i]>large)
            large=a[i];
    }
    cout<<"largest value is"<<large<<endl;
    cout<<"smallest value is:"<<small<<endl;
return 0;
}
```

**Input:**

```
Enter The Array Size:5
ENTER ELEMENTS OF ARRAY5 4 3 2 1
```

**Output:**

```
largest value is5
smallest value is:1
```



**Week-4****4.a) Write a C++ program to sort a list of numbers in ascending order.****Program:**

```
#include<iostream.h>
void sort(int data[],int n)
{
    for(int i=0;i<n;i++)// read the elements of an array
    for(int j=0;j<n-1;j++)
    {
        int t;
        if(data[j]>data[j+1])
        {
            t=data[j];
            data[j]=data[j+1];
            data[j+1]=t;
        }
    }
}
int main()
{
    int a[50],i,n;
    cout<<"Enter How many elements to sort:";
    cin>>n;
    cout<<"Enter Elements:";
    for(i=0;i<n;i++) // read the elements of an array
        cin>>a[i];
    cout<<"Sorted array is \n";
    for(i=0;i<n;i++)
        cout<<a[i]<<"\t";
    return 0;
}
```

**Input:**

Enter How many elements to sort:5

Enter Elements5 4 3 2 1

**Output:**

Sorted array is

5 4 3 2 1

**4.b) Write a Program to illustrate New and Delete Keywords for dynamic memory allocation.****Program:**

```
#include<iostream.h>
int sum(int *a,int n)
{
int s=0;
for(int i=0;i<n;i++)
s=s+*(a+i);
return s;
}
int main()
{
    int *p,i,n;
    cout<<"enter how many values to be read:";
    cin>>n;
    p=new int[n];
    cout<<"Enter values :";
    for(int i=0;i<n;i++)
    cin>>p[i];
    intArray_sum=sum(p,n);
    cout<<"sum of all values are "<<Array_sum;
return 0;
}
```

**Input:**

enter how many values to be read:4

Enter values :1

2

3

4

**Output:**

sum of all values are 10

**Week-5****Write a program Illustrating Class Declarations, Definition, and Accessing Class Members.****Program:**

```
#include<iostream.h>
class sample
{
private:
    int a;
    char b;
    float c;

public:
    voidget_data()
    {
        cout<<"Enter an integer value:";
        cin>>a;
        cout<<"Enter a character:";
        cin>>b;
        cout<<"Enter a float value:";
        cin>>c;
    }
    voidprint_data()
    {
        cout<<"Values read from keyboard are\n";
        cout<<"Integer value:"<<a<<endl;
        cout<<"character is :"<<b<<endl;
        cout<<"float value is :"<<c<<endl;
        cin>>c;
    }
};
int main()
{
    sample s;//creation of object
        s.get_data();
        s.print_data();
}
```

**Output:**

Enter an integer value:12

Enter a character:S

Enter a float value:12.12

Values read from keyboard are

Integer value:12

character is :S

float value is :12.12

**Write a C++ Program to illustrate default constructor,parameterized constructorand copy constructors.**

**Program:**

```
#include<iostream.h>
class code
{
int id;
int count;
public:
    code()
    {
        cout<<"Default constructor called\n";
        id=0;
        cout<<"id="<<id<<endl;
    }
    code(int a)
    {
        cout<<"Parameterized constructor called\n";
        id=a;
        cout<<"id="<<id<<endl;
    }
    code(code&x )
    {
        cout<<"copy constructor called\n";
        id=x.id;
        cout<<"id="<<id<<endl;
    }
    void display()
    {
        cout<<"id="<<id<<endl;
    }
    ~code()
    {
        cout<<"Object Destroyed"<<endl;
    }
}
```

```
    }  
};  
int main()  
{  
code a(100);//calls parameterized constructor  
code b(a); //calls copy constructor  
code c(a); //calls copy constructor  
code d;//calls default constructor  
cout<<"\n For object d id="; d.display();  
cout<<"\n For object a id="; a.display();  
cout<<"\n For object b id="; d.display();  
cout<<"\n For object c id="; d.display();  
return 0;  
}
```

**Output:**

Parameterized constructor called  
id=100  
copy constructor called  
id=100  
copy constructor called  
id=100  
Default constructor called  
id=0  
For object d id=id=0  
For object a id=id=100  
For object b id=id=0For object c id=id=0  
Object Destroyed  
Object Destroyed  
Object Destroyed  
Object Destroyed

**Write a Program to Implement a Class STUDENT having following members:**

Data members	
Member	Description
sname	Name of the student
Marks array	Marks of the student
total	Total marks obtained
Tmax	Total maximum marks

Member functions	
Member	Description
assign()	Assign Initial Values
compute()	to Compute Total, Average
display()	to Display the Data.

**Program:**

```
#include<iostream.h>
#include<string>
class student
{
    charsname[50];
    float marks[6];
    float total;
    floatmax_marks;
public:
    student();
    void assign();
    void compute();
    void display();
};
student::student()
{
    strcpy(sname, " ");
    for(int i=0;i<6;i++)
        marks[i]=0;
```

```
        total=0;
        max_marks=0;
    }
    void student::assign()
    {
        cout<<endl<<"Enter Student Name :";
        cin>>sname;
        for(int i=0;i<6;i++)
        {
            cout<<"Enter marks of"<<i+1<<" subject:";
            cin>>marks[i];
        }
        cout<<"Enter Maximum total marks";
        cin>>max_marks;
    }
    void student::compute()
    {
        total=0;
        for(int i=0;i<6;i++)
            total+=marks[i];
    }
    void student::display()
    {
        cout<<"Student Name:"<<sname<<endl;
        cout<<"Marks are\n";
        for(int i=0;i<6;i++)
            cout<<"Subject "<<i+1<<": "<<marks[i]<<endl;
        cout<<".....\n";
        cout<<"Total :"<<total<<endl;
        cout<<".....\n";
        float per;
        per=(total/max_marks)*100;
```



```
cout<<"Percentage:"<<per;
}

int main()
{
studentobj;
    obj.assign();
    obj.compute();
    obj.display();
return 0;
}
```

**Output:**

```
Enter Student Name :sunil
Enter marks of1 subject:60
Enter marks of2 subject:60
Enter marks of3 subject:65
Enter marks of4 subject:65
Enter marks of5 subject:70
Enter marks of6 subject:75
Enter Maximum total marks600
Student Name:sunil
Marks are
Subject 1: 60
Subject 2: 60
Subject 3: 65
Subject 4: 65
Subject 5: 70
Subject 6: 75
-----
Total    :395
-----
Percentage:65.8333
```

## Week-6

**6. Write a program to demonstrate the i) Operator Overloading ii) Function Overloading.**

**i) Operator Overloading:** -The mechanism of giving a special meaning to an operator is called operator overloading. This can be achieved by special function “**operator**”

Syntax:

```
return type classname:: operator op(list of arguments)
{
.....
}
```

**Program:**

```
#include<iostream.h>
class complex
{
float real,img;
public:
    complex();
    complex(float x,float y);
    void read_complex();
    complex operator+(complex);
    complex operator-(complex);
    void display();
};
complex::complex()
{
    real=img=0;
}
complex::complex(float x,float y)
{
    real=x;
    img=y;
}
void complex::display()
{
char sign;
    if(img<0)
    {
        sign='-';
        img=-img;
    }
    else
    {
        sign='+';
    }
}
```

```
        }
        cout<<real<<sign<<"i"<<img<<endl;
    }
    complex complex::operator+(complex c)
    {
    complex r;
        r.real=real+c.real;
        r.img=img+c.img;
    return r;
    }
    complex complex::operator-(complex c)
    {
    complex r;
        r.real=real-c.real;
        r.img=img-c.img;
    return r;
    }
    void complex::read_complex()
    {
        cout<<"Enter real part of complex number;";
        cin>>real;
        cout<<"Enter Imaginary part of complex number:";
        cin>>img;
    }
    int main()
    {
    complex a;
    a.read_complex();
    complex b;
    b.read_complex();
    complex c;
    c=a+b;
    cout<<"After Addition of two complex numbers";
    c.display();
    c=a-b;
    cout<<"Difference of two complex numbers";
    c.display();
    }
```

Output:

```
Enter real part of complex number;1
Enter Imaginary part of complex number:2
Enter real part of complex number;2
Enter Imaginary part of complex number:4
After Addition of two complex numbers3+i6
```

Difference of two complex numbers-1-i2

### ii)Function Overloading

```
#include<iostream>
usingnamespacestd;
classprintData
{
public:
voidprint(int i)
{
cout<<"Printing int: "<< i <<endl;
}
voidprint(double f)
{
cout<<"Printing float: "<< f <<endl;
}
voidprint(char*c)
{
cout<<"Printing string: "<< c <<endl;
}
};
int main(void)
{
printDatapd;
// Call print to print integer
pd.print(5);
// Call print to print float
pd.print(500.263);
// Call print to print character
pd.print("Hello C++");
return0;
}
```

### Output:

Printingint:5

Printingfloat:500.263

Printing string:Hello C++

**7. b) Write a Program to demonstrate friend function and friend class.****Program:**

```
#include<iostream>
using namespace
std; class sample2;
class sample1
{
int x;
public
:
    sample1(int a);
    friend void max(sample1 s1,sample2 s2);
};
sample1::sample1(int a)
{
x=a;
}
class sample2
{
int y;
public
:
    sample2(int b);
    friend void max(sample1 s1,sample2 s2);
};

sample2::sample2(int b)
{
    y=b;
}

void max(sample1 s1,sample2 s2)
{
if(s1.x>s2.y)
    cout<<"Data member in Object of class sample1 is larger "<<endl;
els
e    cout<<"Data member in Object of class sample2 is larger "<<endl;
}
int main()
{
sample1
obj1(3);
sample2
obj2(5);
    max(obj1,obj2);
}
Output
```

Data member in Object of class sample2 is larger



**Week-7**

**7. a) Write a program to access members of a STUDENT class using pointer to object members.**

**Program:**

```
#include<iostream.h>
class student
{
introllno;
char name[50];
public:
voidgetdata();
void print();
};
void student::getdata()
{
cout<<"Enter roll number"<<endl;
cin>>rollno;
cout<<"Enter Name ";
cin>>name;
}

void student::print()
{
cout<<"Name :"<<name<<endl;
cout<<"Roll no:"<<rollno<<endl;
}
int main()
{
student a;
a.getdata();
a.print();
cout<<"Pointer to class\n";
student *ptr;
ptr=&a;
ptr->print();
}
```

Output:

```
Enter roll number
123
Enter Name jayapal
Name :jayapal
Roll no:123
Pointer to class
Name :jayapal
Roll no:123
```

**7. b) Write a Program to generate Fibonacci Series by using Constructor to initialize the Data Members.**

**Program:**

```
#include<iostream>
using namespace std;
class fibonacci{
int f0,f1,f;
public:
    fibonacci()
    {

        f0=0;
        f1=1;
    }
    void series(int n)
    {
        int count=0;
        f0=0;
        f1=1;
        while(count<n)
        {
            cout<<f0<<"\t";
            count++;
            f=f0+f1;
            f0=f1;
            f1=f;
        }
    }
};
int main()
{
    fibonacci obj;
    int terms;
    cout<<"Enter How many terms to be printed:";
    cin>>terms;
    obj.series(terms);

    return 0;
}
```

**Output:** Enter How many terms to be printed:5

0 1 1 2 3

**Week-8**  
**Revision Of Programs**



## Week-9

9) Write a c++ program to implement the matrix ADT using a class. The operations supported by this ADT are:

- a) Reading a matrix    b) addition of matrices    c) printing a matrix  
d) subtraction of matrices    e) multiplication of matrices

**Program:**

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
#include<iomanip.h>
class matrix
{
protected:
inti,j,a[10][10],b[10][10],c[10][10];
int m1,n1,m2,n2;
public:
virtual void read()=0;
virtual void display()=0;
virtual void sum()=0;
virtual void sub()=0;
virtual void mult()=0;

};
classresult:public matrix
{
public:
void read();
void sum();
void sub();
voidmult();
void display();
};
void result::read()
{
cout<<"\nenter the order of matrix A ";
cin>>m1>>n1;
cout<<"\nenter the elements of matrix A ";
```

```
        for(i=0;i<m1;i++)
        {
for(j=0;j<n1;j++)
        {
cin>>a[i][j];
        }
        }
cout<<"\nenter the order of matrix B ";
cin>>m2>>n2;
cout<<"\nenter the matrix B ";
for(i=0;i<m2;i++)
        {
for(j=0;j<n2;j++)
        {
cin>>b[i][j];
        }
        }
}
void result::display()
{

for(i=0;i<m1;i++)
        {
for(j=0;j<n1;j++)
        {
cout.width(3);
cout<<c[i][j];
        }
cout<<"\n";
        }
}
void result::sum()
{
if((m1!=m2)||(n1!=n2))
        {
cout<<"the order should be same for addition";
        }
else
        {
```

```
for(i=0;i<m1;i++)
    {
for(j=0;j<n1;j++)
    {
c[i][j]=a[i][j]+b[i][j];
    }
    }
}
void result::sub()
    {
if((m1!=m2)||(n1!=n2))
    {
cout<<"the order should be same for subtraction ";
    }
else
    {
for(i=0;i<m1;i++)
    {
for(j=0;j<n1;j++)
    {
c[i][j]=a[i][j]-b[i][j];
        //cout<<a[i][j];
    }
    }
    }
}
void result::mult(void)
    {
if(n2!=m2)
    {
cout<<"Invalid order limit ";
    }
else
    {
for(i=0;i<m1;i++)
    {
for(j=0;j<n2;j++)
    {
c[i][j]=0;
```

```

for(int k=0;k<n1;k++)
    {
c[i][j]+=a[i][k]*b[k][j];
    }
    }
}
}

void main()
{
intch;
class matrix *p;
class result r;
    p=&r;
clrscr();
while(1)
    {

cout<<"\n1. Addition of matrices ";
cout<<"\n2. Subtraction of matrices ";
cout<<"\n3. Multiplication of matrices ";
cout<<"\n4. Exit";
cout<<"Enter your choice ";
cin>>ch;
switch(ch)
    {
case 1:
p->read();
p->sum();
p->display();
break;
case 2:
    (p)->read();
p->sub();
p->display();
break;
case 3:
p->read();
p->mult();

```

```
p->display();
break;
case 4:
exit(0);
    }
    }
}
```

**Output:**

1. Addition of matrices
2. Subtraction of matrices
3. Multiplication of matrices
4. Exit

Enter your choice

1

enter the order of matrix A

2 2

enter the elements of matrix A

1 1

1 1

enter the order of matrix B

2 2

enter the elements of matrix B

1 1

1 1

2 2

2 2

## Week-10

**10.a) Write a C++ Program that illustrate single inheritance.**

The mechanism of deriving a new class from an old one is called **inheritance** or **derivation**

```
class derived-class-name : visibility-mode base-class-name
{
.....
.....
}
```

**Program:**

```
#include<iostream>
using namespace
std; class A
{
protected:
    int a,b;
    public
    :
    void get()
    {
        cout<<"Enter any two integer
        values"; cin>>a>>b;
    }
};
class B:public A
{
    int c;
    public
    :
    void add()
    {
        c=a+b;
        cout<<a<<"+"<<b<<"="<<c;
    }
};
int main()
{
    B b;
    b.get()
    ;
    b.add();
}
```

**Output:**

```
Enter any two integer values
1
2
1+2=3
```

**10.b) Write a C++ Program that illustrate multiple inheritance.****Program:**

```

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

class student
{
protected:
    int rno,m1,m2;
public:
    void get()
    {
        cout<<"Enter the Roll no :";
        cin>>rno;
        cout<<"Enter the two marks :";
        cin>>m1>>m2;
    }
};

class sports
{
protected:
    intsm;          // sm = Sports mark
public:
    voidgetsm()
    {
        cout<<"\nEnter the sports mark :";
        cin>>sm;
    }
};

classstatement:publicstudent,public sports
{
    inttot,avg;
public:
    void display()
    {
        tot=(m1+m2+sm);
        avg=tot/3;
        cout<<"\n\n\tRoll No   : "<<rno<<"\n\tTotal   : "<<tot;
    }
};

```

```
        cout<<"\n\tAverage  : "<<avg;
    }
};
void main()
{
    clrscr();
    statementobj;
    obj.get();
    obj.getsm();
    obj.display();
    getch();
}
```

**Output:**

Enter the Roll no: 100

Enter two marks

90

80

Enter the Sports Mark: 90

Roll No: 100

Total : 260

Average: 86.66



**10.c) Write a C++ Program that illustrate multi level inheritance.****Program:**

```
#include<iostream.h>
#include<conio.h>
class top          //base class
{
public :
int a;
voidgetdata()
{
cout<<"\n\nEnter first Number :::\t";
cin>>a;
}
voidputdata()
{
cout<<"\nFirst Number Is :::\t"<<a;
}
};

//First level inheritance
class middle :public top    // class middle is derived_1
{
public:
int b;
void square()
{
getdata();
b=a*a;
cout<<"\n\nSquare Is ::: "<<b;
}
};

//Second level inheritance
class bottom :public middle // class bottom is derived_2
{
public:
int c;
void cube()
{
square();
c=b*a;
cout<<"\n\nCube :::\t"<<c;
}
};
```

```
int main()
{
    clrscr();
    bottom b1;
    b1.cube();
    getch();
}
```

**Input:**

Enter first number ::: 4

**Output:**

Square Is ::: 16

Cube ::: 64

**10.d) Write a C++ Program that illustrate Hierarchical inheritance.****Program:**

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

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

```
class A //Base Class
```

```
{
    public:
    int a,b;
    void getnumber()
    {
        cout<<"\n\nEnter Number :::\t";
        cin>>a;
    }
};
```

```
class B : public A //Derived Class 1
```

```
{
    public:
    void square()
    {
        getnumber(); //Call Base class property
        cout<<"\n\n\tSquare of the number :::\t"<<(a*a);
        cout<<"\n\n\t..... ";
    }
};
```

```
class C :public A //Derived Class 2
```

```
{
    public:
    void cube()
    {
        getnumber(); //Call Base class property
        cout<<"\n\n\tCube of the number :::\t"<<(a*a*a);
        cout<<"\n\n\t..... ";
    }
};
```

```
int main()
{
clrscr();
B b1;    //b1 is object of Derived class 1
b1.square(); //call member function of class B
C c1;    //c1 is object of Derived class 2
c1.cube(); //call member function of class C
getch();
}
```

**Input:**

Enter number ::: 2

**Output:**

Square of the number ::: 4

---

**Input:**

Enter number ::: 2

**Output:**

Cube of the number ::: 8

---

**Week-11**

**Write a C++ program to illustrate the order of execution of constructors and destructors.**

**Program:**

```
#include<iostream.h>
class Base
{
public:

    Base ()
    {
cout<< "Inside Base constructor" <<endl;
    }
~Base ()
{
cout<< "Inside Base destructor" <<endl;
    }
};
class Derived : public Base
{
public:
Derived ()
{
cout<< "Inside Derived constructor" <<endl;
    }
~Derived ()
{
cout<< "Inside Derived destructor" <<endl;
    }
};

void main( )
{
    Derived x;
}
```

**Output:**

Inside Base constructor

Inside Derived constructor

Inside Derived destructor

Inside Base destructor

**write a program to invoking derived class member through base class pointer.**

**Program:**

```
#include <iostream.h>
#include <conio.h>
class A
{
public:
    virtual void print_me(void)
    {
        cout<< "I'm A" <<endl;
    }
virtual ~A()
    {
    }
};
class B : public A
{
public:
    virtual void print_me(void)
    {
        cout<< "I'm B" <<endl;
    }
};
class C : public A
{
public:
    virtual void print_me(void)
    {
        cout<< "I'm C" <<endl;
    }
};
int main()
{
    A a;
    B b;
    C c;
    clrscr();
    A* p = &a;
    p->print_me();
```

```
p = &b;  
p->print_me();  
    p = &c;  
p->print_me();  
return 0;  
}
```

**Output:**

I'm A  
I'm B  
I'm C



**Week-12**

**12.a)Write a template based program to sort the given list of elements.**

**Program:**

```
#include<iostream.h>
using namespace std;

template<class T>
void bubble(T a[], int n)
{
    int i, j;
    for(i=0;i<n-1;i++)
        {
            for(j=0;j<n-1;j++)
                {
                    if(a[j]>a[j+1])
                        {
                            T temp;
                            temp = a[j];
                            a[j] = a[j+1];
                            a[j+1] = temp;
                        }
                }
        }
}

int main()
{
    int a[6]={ 17,16,15,14,9,-1 };
    char b[4]={'z','b','x','a'};

    bubble(a,6);
    cout<<"\nSorted Order Integers: ";
    for(int i=0;i<6;i++)
        cout<<a[i]<<"\t";
    bubble(b,4);

    cout<<"\nSorted Order Characters: ";
    for(int j=0;j<4;j++)
        cout<<b[j]<<"\t";

}
```

**Output:**

```
Sorted Order Integers: -1    9    14    15    16    17
Sorted Order Characters: a    b    x    z
```

**12.b) Write a C++ program that uses function templates to find the largest and smallest number in a list of integers and to sort a list of numbers in ascending order.**

**Program:**

```
#include<iostream.h>
template<class T>          //Template declaration
voidmaxmin(T a[],int n)   //Function Template
{
int i;
    T temp;
for(i=0;i<n;i++)
for(int j=i+1;j<n;j++)
    {
if(a[i]>a[j])
    {
temp=a[i];
a[i]=a[j];
a[j]=temp;
    }
    }
cout<<"max="<<a[n-1]<<"\n"<<"min="<<a[0]<<"\n";
    /*After sorting an Array starting index consists of Small element and Final index consists of
Largest element */
cout<<"sorted list is: \n";
for(i=0;i<n;i++)
cout<<a[i]<<" ";
}

int main()
{
int a[50],i,ch,n;
double d[50];
float f[50];
char c[50];
cout<<"1.integer"<<endl;
cout<<"2.characters"<<endl;
cout<<" 3.float numbers"<<endl;
cout<<" 4.double numbers"<<endl;
cout<<"enter corresponding Index Example : enter '1' for integers"<<endl;
```

---

```
cin>>ch;           //Reading Choice from User
cout<<"enter the n value\n";
cin>>n;           //Number of elements is independent of DATA TYPE
switch(ch)
{
case 1:           //for operations over Integer Array
cout<<"enter integers\n";
for(i=0;i<n;i++)
cin>>a[i];
maxmin(a,n);
break;

case 2:           //for operations over Character Array
cout<<"enter characters\n";
for(i=0;i<n;i++)
cin>>c[i];
maxmin(c,n);
break;

case 3:           //for operations over Floating Array
cout<<"enter floatnumbers\n";
for(i=0;i<n;i++)
cin>>f[i];
maxmin(f,n);
break;

case 4:           //for operations over Double
cout<<"enter doublenegers\n";
for(i=0;i<n;i++)
cin>>d[i];
maxmin(d,n);
break;

default:
cout<<"Invalid choice entered...";
}
return 0;
}
```

---

## Week-13

**Write a C++ program containing a possible exception.use a try block to throw it and a catch block to handle it properly.**

### Program:

```
#include <iostream>
using namespace std;
int main()
{
int x = -1;
cout<< "Before try \n";
try
{
cout<< "Inside try \n";
if (x < 0)
{
throw x;
cout<< "After throw (Never executed) \n";
}
}
catch (int x )
{
cout<< "Exception Caught \n";
}
cout<< "After catch (Will be executed) \n";
return 0;
}
```

### Output:

```
Before try
Inside try
Exception Caught
After catch (Will be executed)
```

**Write a C++ program to demonstrate the catching of all exceptions.**

**Program:**

```
#include<iostream.h>
#include<conio.h>
void test(int x)
{
    try
    {
        if(x>0)
            throw x;
        else
            throw 'x';
    }
}
catch(int x)
{
    cout<<"Catch a integer and that integer is:"<<x;
}
catch(char x)
{
    cout<<"Catch a character and that character is:"<<x;
}
}
void main()
{
    clrscr();
    cout<<"Testing multiple catches\n";
    test(10);
    test(0);
    getch();
}
```

**Output:**

Testing multiple catches

Catch a integer and that integer is: 10

Catch a character and that character is: x