Department of Computer Science

St. Joseph's Degree & P.G College

(Autonomous), Affiliated to Osmania University
Re-accredited by NAAC with A Grade with CGPA 3.49
A Catholic Christian Minority Institution
King Koti Road, Hyderabad.

Lab Manual

Program	: B.Sc [MPCs/MSCs/MECs] I Year Semester I
Course	: OOP's Using C++
Course code	: BS.05.203.11P

B.Sc (MPCs/MECs/MSCs)

I Year / I Semester

THEORY PAPER – I

Object Oriented Programming Using C++
(w.e.f 2019-20)

Scheme of Instruction	**Scheme of Examination**
Total durations Hrs : 60	Max. Marks : 100
Hours/Week : 06 (4T+2P)	Internal Examination :30
Credits : 5	SBT : 10
Instruction Mode: Lecture + Practical	External Examinatio:60
Course Code : BS.05.201.13.T	Exam Duration : 3 Hrs

Course Objectives:

To impart students with knowledge on basics of programming and Object Oriented Programming concepts.

Course Outcomes:

At the end of the course the student will be able to

CO 1: Understand the basics of programming and develop simple programs in C++ using Controlstructures.

CO2: Understand the concepts of Arrays, Pointers, Functionsand perform Modular Programming.

CO 3: Acquire knowledge on Object Oriented Programming Concepts and design programs using Constructors, friend functions and templates.

CO 4: Develop Software Applications using the concepts like Polymorphism, Inheritance and Exceptional Handling mechanisms.

UNIT-1: Programming Concepts and C++Basics

Programming Concepts: Program, Structured Programming, Object Oriented Programming.

C++ Basics: Introduction to C++, Layout of C++ program, Data types, variables, constants, Keywords, Operators.

Control statements: Branching Statements: if, if-else, nested if, Break,continue and switch statement. Looping Statements: While, Do-while and for Statement.

UNIT-2: Arrays, Pointers and Functions.

Arrays: Introduction, One-dimensional Arrays-Declaration, Initialization, Two-dimensional Arrays-Declaration, Initialization.

Pointers: Introduction, Uses of pointer, Declaring Pointer Variables, Initialization of Pointer Variables, Accessing a Variable through its Pointer.

Functions: Introduction, definition of function, Built-in functions, User defined functions: Elements of Functions, Parameter Passing and Recursive Functions.

UNIT-3: Objects, Classes and Templates.

Object & Classes: Features of Object Oriented Programming, Class specification, Access Specifiers, Defining Member Functions, Objects Declaration, Accessing Data Members and Member Functions, Constructors, Destructor, Friend Functions.

Templates: Function Templates, Class Templates

UNIT-4: Inheritance,Polymorphism and Exception Handling

Inheritance: Introduction to inheritance, Base Class, Derived class, Types of Inheritance,

Polymorphism: Function Overloading, Function Overriding, Virtual Functions,

Operator Overloading.

Exception Handling: Introduction, Exception Handling Mechanism, Handling Multiple Exceptions.

Text Book:

Mastering C++ by R Venugopal, Rajkumar& T Ravishankar, Tata McGrawHill

References:

1. Tony Gaddis, Starting out with C++: from control structures through objects (7e)

2. B. Stroupstrup, The C++ Programming Language, Addison Wesley, 2004.
3. Problem Solving with C++ by Walter Savitch, Addison Wesley

B.Sc. (MSCs/MPCs/MECs)
I Year / I Semester
PRACTICAL PAPER - I
Object Oriented Programming using C++

Subject Code: BS.05.201.11.P

Course Objectives:
To impart the aspirants about the basics of programming concepts, OOPs concepts and OOD concepts.

Course Outcomes:
On successful completion of this course, the students should have

- Gained the practical knowledge on basics and implentationof C++ programming concepts.
- Students can gain knowledge in demonstrating different functions.

St. Joseph’s Degree &amp; PG College

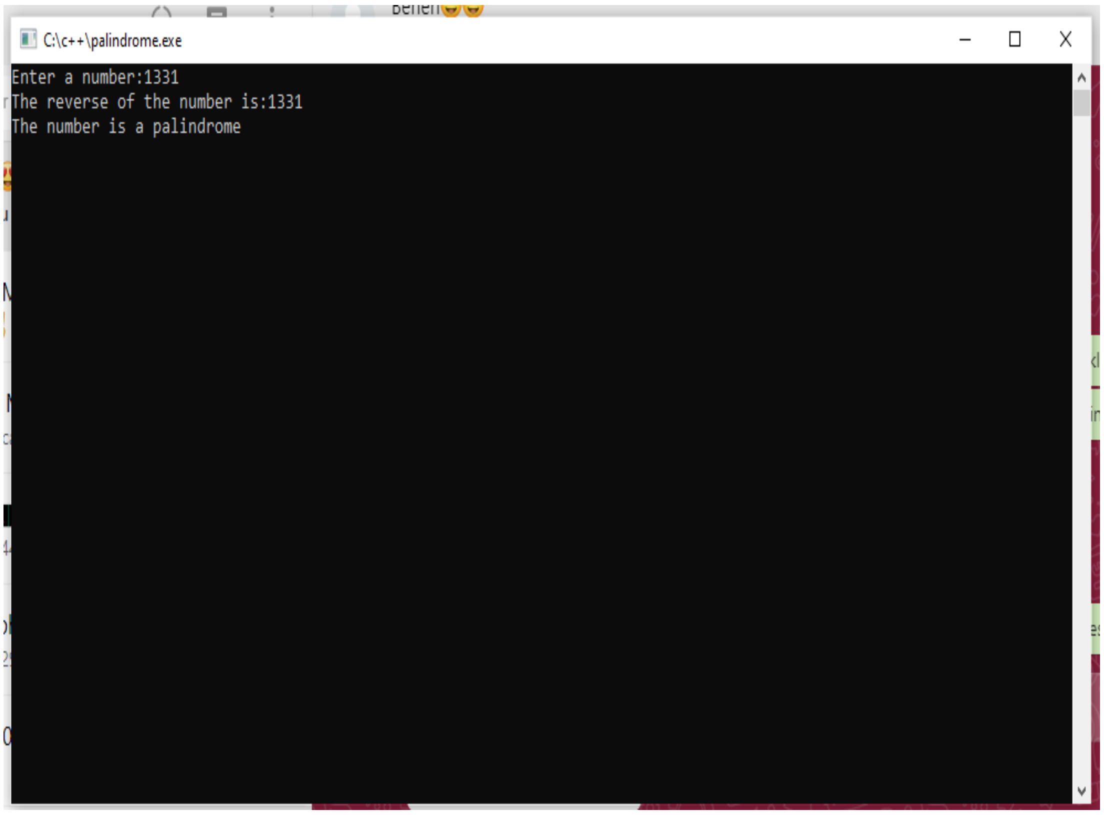
Department of Computer Science

OOPs Using C++ Semester- I Lab Manual

S no	Name of the programs
1	Write a c++ program to check whether given number is Palindrome or not.
2	Write a c++ Program to implement Matrices multiplication
3	Write a c++ program to implement Functions
4	Write a c++ program to implement Bank account Class
5	Write a c++ program to implement Student information Class
6	Write a c++ program to implement Constructors.
7	Write a c++ program to find Factorial of a given number using Recursion
8	Write a c++ program to implement Friend function
9	Write a c++ program to implement Function Templates
10	Write a c++ program to implement Multiple inheritance
11	Write a c++ program to implement Hierarchical inheritance
12	Write a c++ program to implement Function overloading
13	Write a c++ program to implement Exceptional handling
14	Write a c++ program to implement Class Templates.
15	Write a c++ program to implement Virtual Functions

1) Program of palindrome

```
#include<iostream.h>
#include<conio.h>
int main()
{
        int n, num, digit ,rev=0;
        cout<<"Enter a number:";
        cin>>num;
        n=num;
        while(num>0)
        {
                digit=num%10;
                rev=(rev*10)+digit;
                num=num/10;
        }
        cout<<"The reverse of the number is:"<<rev<<endl;
        if(n==rev)
        cout<<"The number is a palindrome";
        else
                {
                cout<<"The number is not a palindrome";
        }
        getch();
        return 0;
}
```


```
Enter a number:1331
The reverse of the number is:1331
The number is a palindrome
```

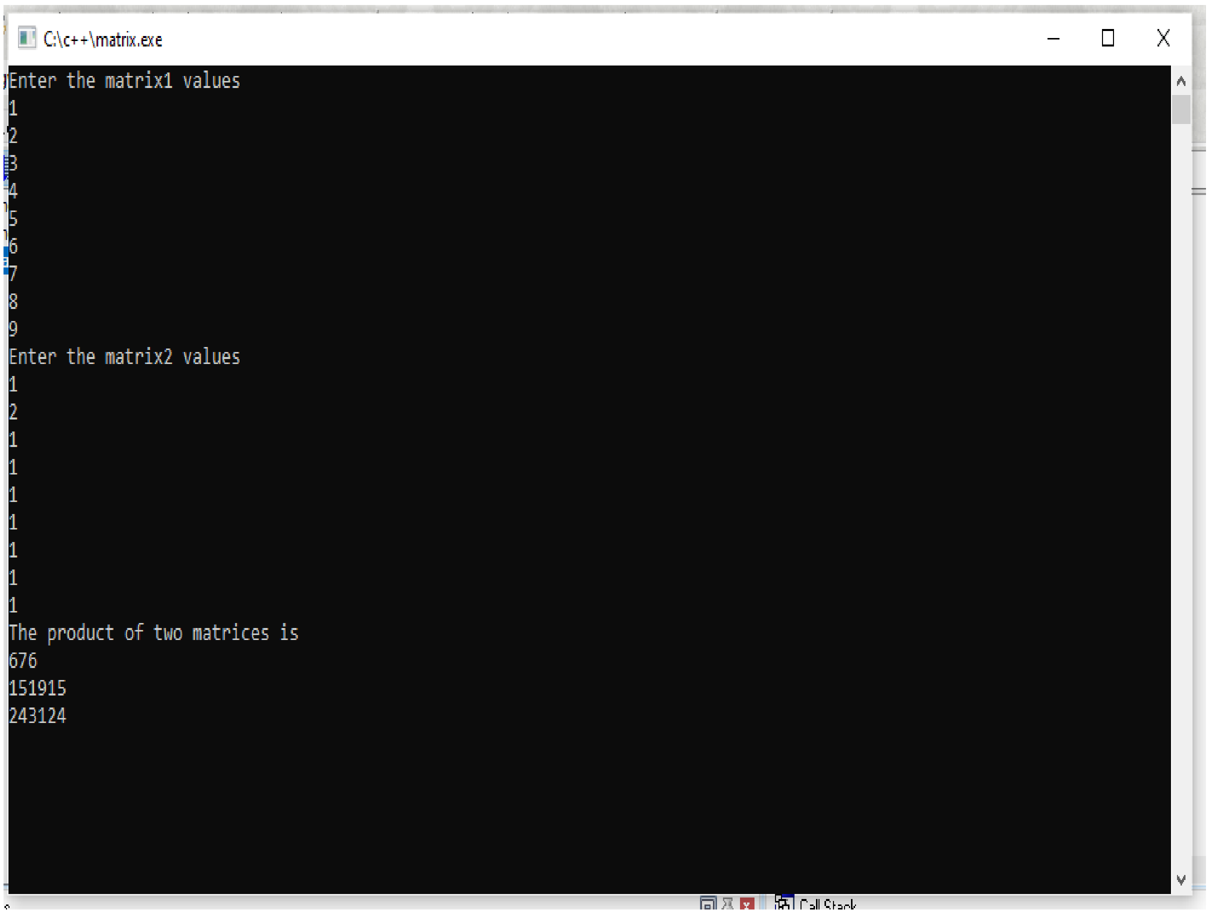
Output of palindrome program

2) Program on product of matrices

```
#include<iostream.h>
#include<conio.h>
int main()
{
	int a[3][3], b[3][3], c[3][3], i, j, k;
	cout<<"Enter the matrix1 values"<<endl;
	for(i=0; i<3;i++)
	{
		for(j=0; j<3; j++)
		{
			cin>>a[i][j];
		}
	}
	cout<<"Enter the matrix2 values"<<endl;
```

```
for(i=0; i<3; i++)
{
        for(j=0; j<3; j++)
        {
                cin>>b[i][j];
        }
}
for(i=0; i<3; i++)
{
        for(j=0; j<3; j++)
        {
                c[i][j]=0;
                for(k=0; k<3; k++)
                {
                        c[i][j]=c[i][j]+a[i][k]*b[k][j];
                }
        }
}
cout<<"The product of two matrices is"<<endl;
for(i=0; i<3; i++)
{
        for(j=0; j<3; j++)
        {
                cout<<c[i][j]<<" ";
        }
        cout<<endl;
}
```

```
getch();
return 0;
}
```



C:\c++\matrix.exe

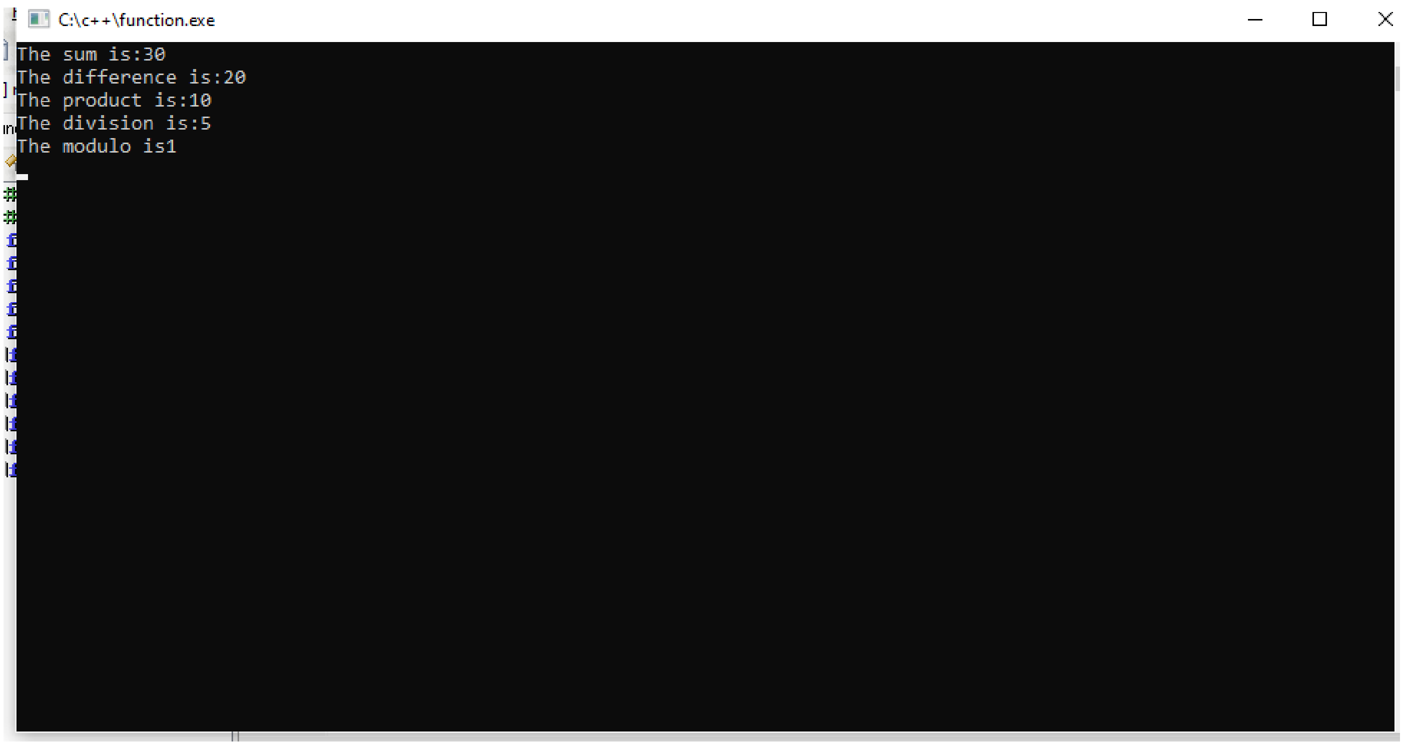
```
Enter the matrix1 values
1
2
3
4
5
6
7
8
9
Enter the matrix2 values
1
2
1
1
1
1
1
1
1
The product of two matrices is
676
151915
243124
```

Output of product matrix program

3) <u>Program on functions</u>

```
#include<iostream.h>
#include<conio.h>
void add(int a,int b);
void sub(int a,int b);
void mul(int a,int b);
void divide(int a,int b);
void modulo(int a,int b);
int main()
{
        add(20, 10);
        sub(50, 30);
        mul(2, 5);
        divide(50, 10);
        modulo(25, 4);
}
void add(int a,int b)
{
        int c=a+b;
        cout<<"The sum is:"<<c<<endl;
}
void sub(int a,int b)
{
        int c=a-b;
        cout<<"The difference is:"<<c<<endl;
}
void mul(int a,int b)
{
        int c=a*b;
        cout<<"The product is:"<<c<<endl;
}
void divide(int a,int b)
{
        int c=a/b;
        cout<<"The division is:"<<c<<endl;
}
void modulo(int a,int b)
```

```
{
        int c=a%b;
        cout<<"The modulo is"<<c<<endl;
        getch();
}
```



```
The sum is:30
The difference is:20
The product is:10
The division is:5
The modulo is1
```

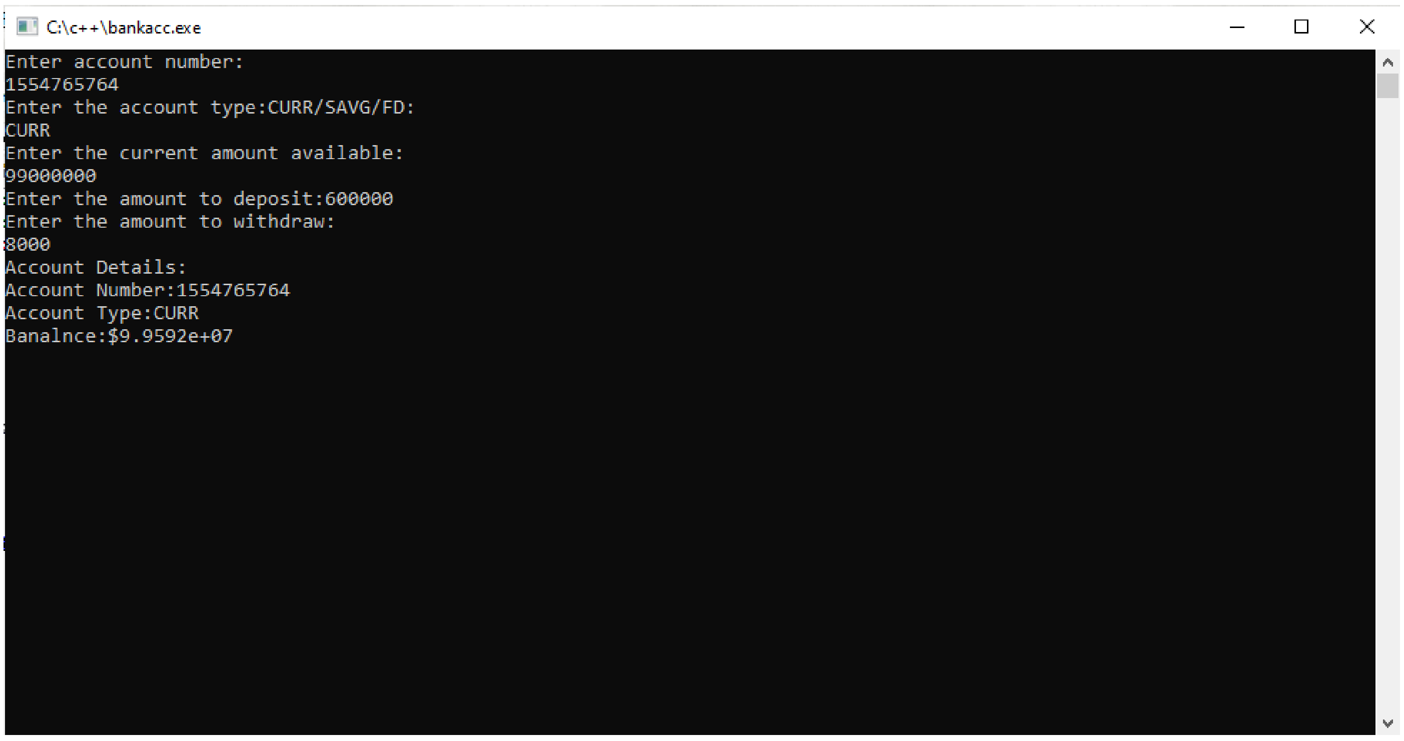
Output of functions program

---

4) <u>Program of bank account class</u>

```
#include<iostream.h>
#include<conio.h>
class BankAccount
{
        int acno;
        float balance;
        char actype[4];
        public:
        void store();
        void deposit();
        void withdraw();
        void display();
};
void BankAccount :: store()
{
        cout<<"Enter account number:"<<endl;
        cin>>acno;
        cout<<"Enter the account type:CURR/SAVG/FD:"<<endl;
        cin>>actype;
        cout<<"Enter the current amount available:"<<endl;
        cin>>balance;
        }
        void BankAccount :: deposit()
        {
                float more;
                cout<<"Enter the amount to deposit:";
                cin>>more;
                balance = balance + more;
        }
        void BankAccount :: withdraw()
        {
                float amt;
                cout<<"Enter the amount to withdraw:"<<endl;
                cin>>amt;
                balance = balance - amt;
        }
```

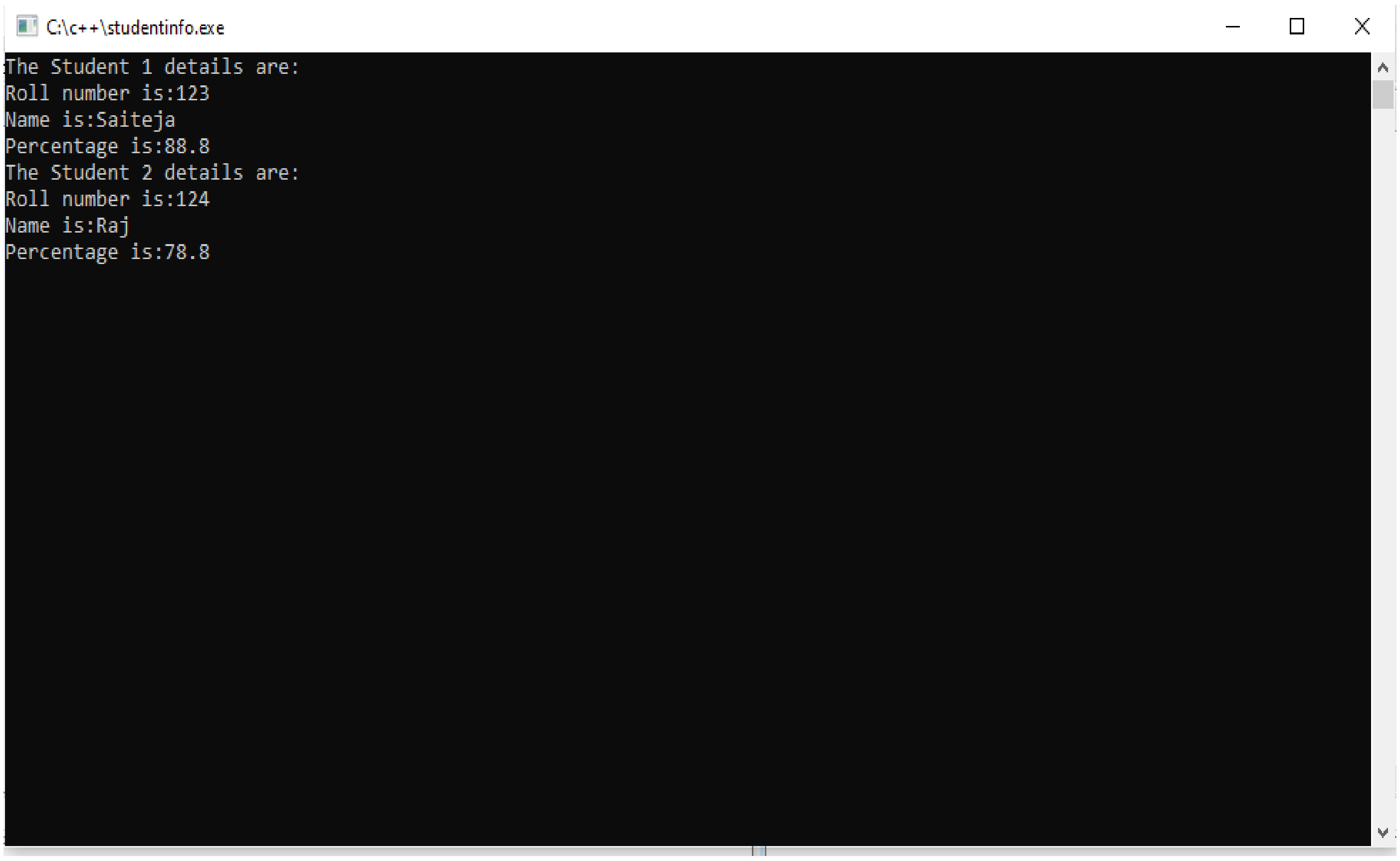
```
void BankAccount :: display()
{
        cout<<"Account Details:"<<endl;
        cout<<"Account Number:"<<acno<<endl;
        cout<<"Account Type:"<<actype<<endl;
        cout<<"Banalnce:$"<<balance;
}
int main()
{
        BankAccount b;
        b.store();
        b.deposit();
        b.withdraw();
        b.display();
        getch();
        return 0;
}
```



```
Enter account number:
1554765764
Enter the account type:CURR/SAVG/FD:
CURR
Enter the current amount available:
99000000
Enter the amount to deposit:600000
Enter the amount to withdraw:
8000
Account Details:
Account Number:1554765764
Account Type:CURR
Banalnce:$9.9592e+07
```

5) Program on student information

```
#include<iostream.h>
#include<conio.h>
class Student
{
        int rno;
        float percentage;
        char *name;
        public:
        void store(int a,float b,char *c)
        {
                rno = a;
                percentage = b;
                name = c;
        }
        void display()
        {
                cout<<"Roll number is:"<<rno<<endl;
                cout<<"Name is:"<<name<<endl;
                cout<<"Percentage is:"<<percentage<<endl;
        }
};
int main()
{
        Student s1, s2;
        cout<<"The Student 1 details are:"<<endl;
        s1.store (123, 88.8, "Saiteja");
        s1.display();
        cout<<"The Student 2 details are:"<<endl;
        s2.store (124, 78.8, "Raj");
        s2.display();
        getch();
        return 0;
}
```



```
The Student 1 details are:
Roll number is:123
Name is:Saiteja
Percentage is:88.8
The Student 2 details are:
Roll number is:124
Name is:Raj
Percentage is:78.8
```

Output of student information program

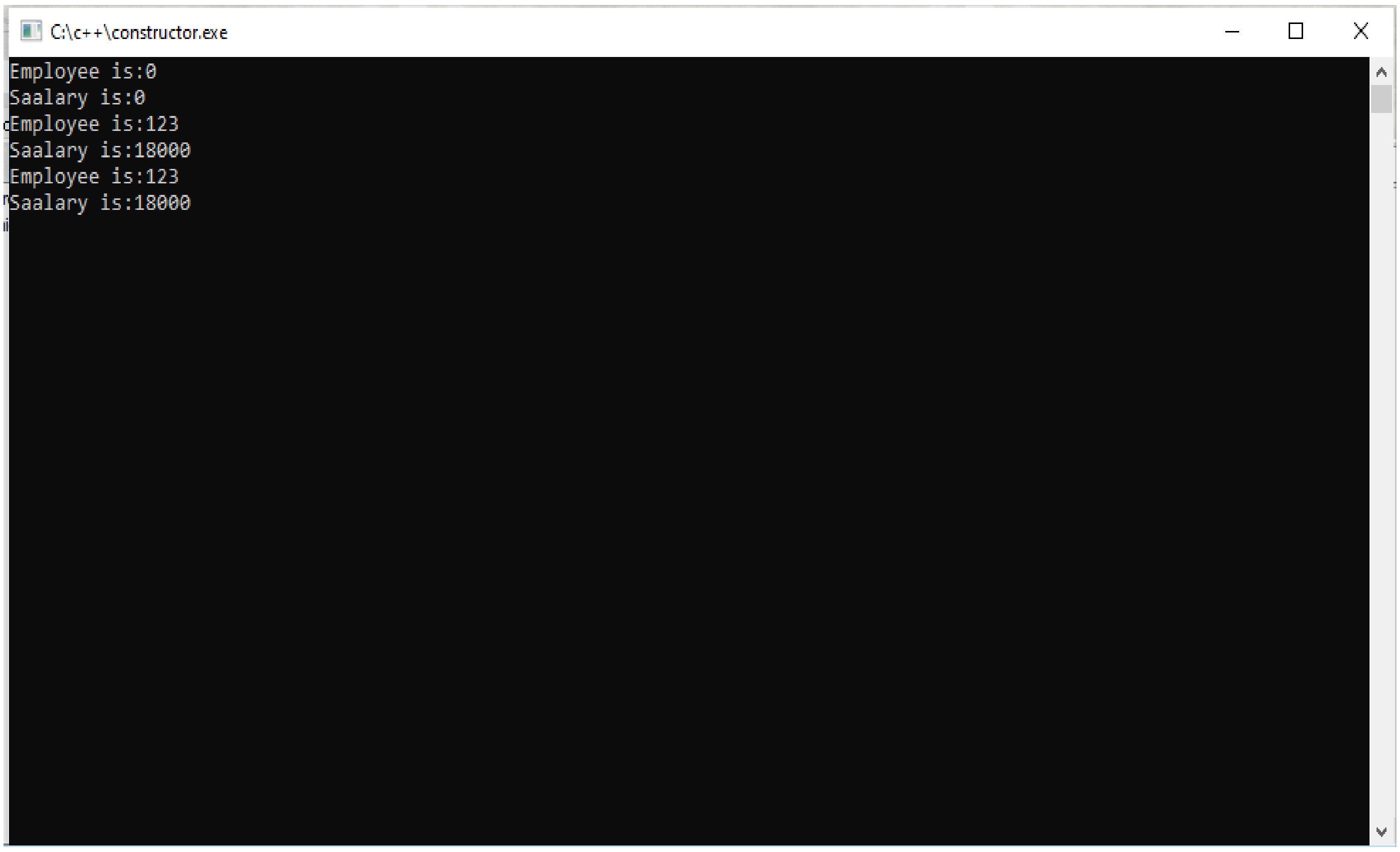
---

6) Program on constructors

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

```
class employee
{
        int eid;
        float salary;
        public:
        employee()
        {
                eid = 0;
                salary = 0.0;
        }
        employee (int x,float y)
        {
                eid = x;
                salary = y;
        }
        employee (employee &e)
        {
                eid = e.eid;
                salary = e.salary;
        }
        void display()
        {
                cout<<"Employee is:"<<eid<<endl;
                cout<<"Saalary is:"<<salary<<endl;
        }
};
int main()
```

```
{
        employee e1;
        e1.display();
    employee e2(123, 18000);
    e2.display();
    employee e3(e2);
    e3.display();
    getch();
    return 0;
}
```



```
Employee is:0
Saalary is:0
Employee is:123
Saalary is:18000
Employee is:123
Saalary is:18000
```

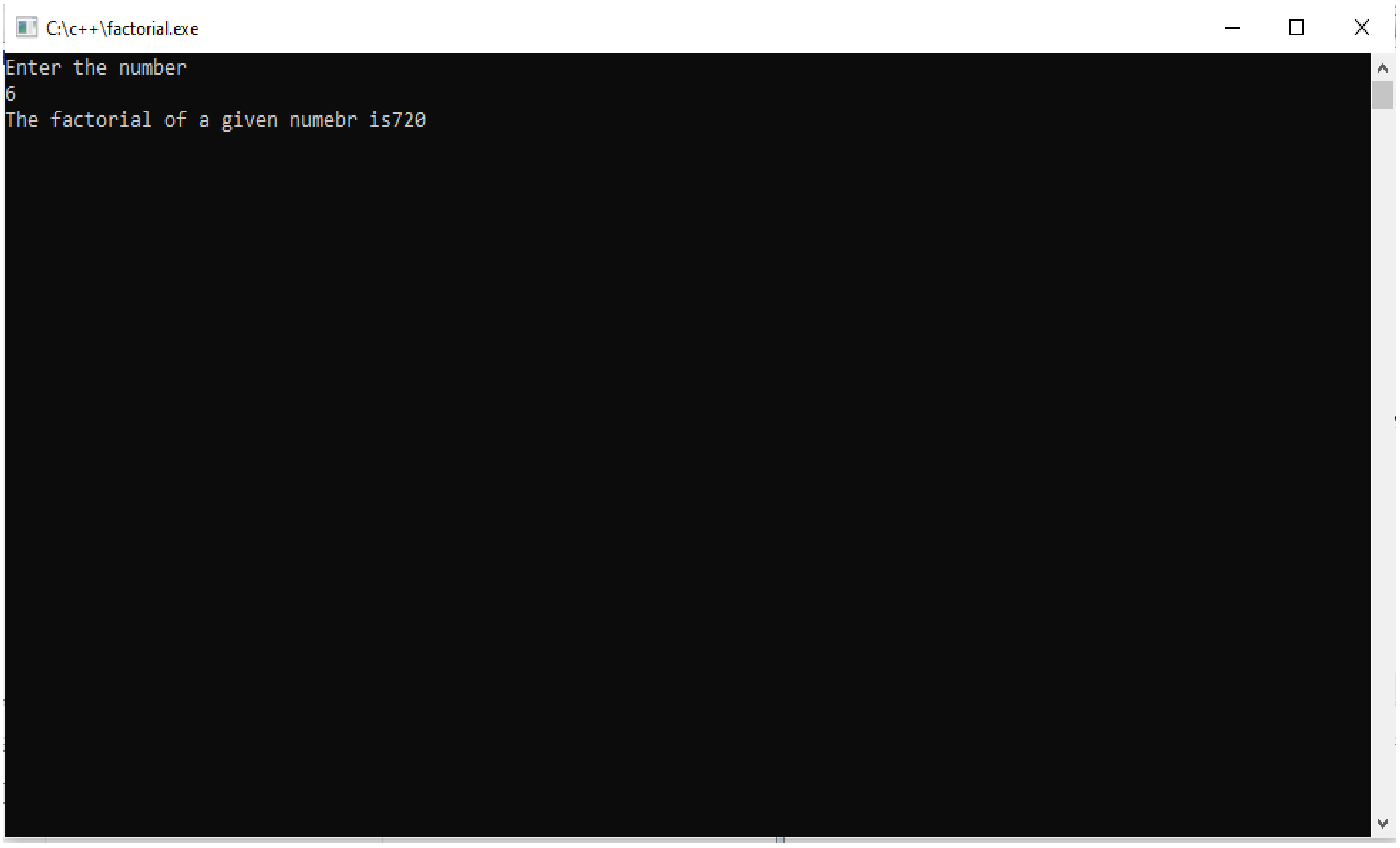
Output of constructor program

---

7) Program on factorial

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

```
int fact(int n);
int main()
{
        int n;
        cout<<"Enter the number"<<endl;
        cin>>n;
        cout<<"The factorial of a given 14umber is"<<fact(n)<<endl;
        getch();
        return 0;
}
int fact(int n)
{
        if (n==0||n==1)
        return 1;
        else
        return n* fact(n-1);
}
```

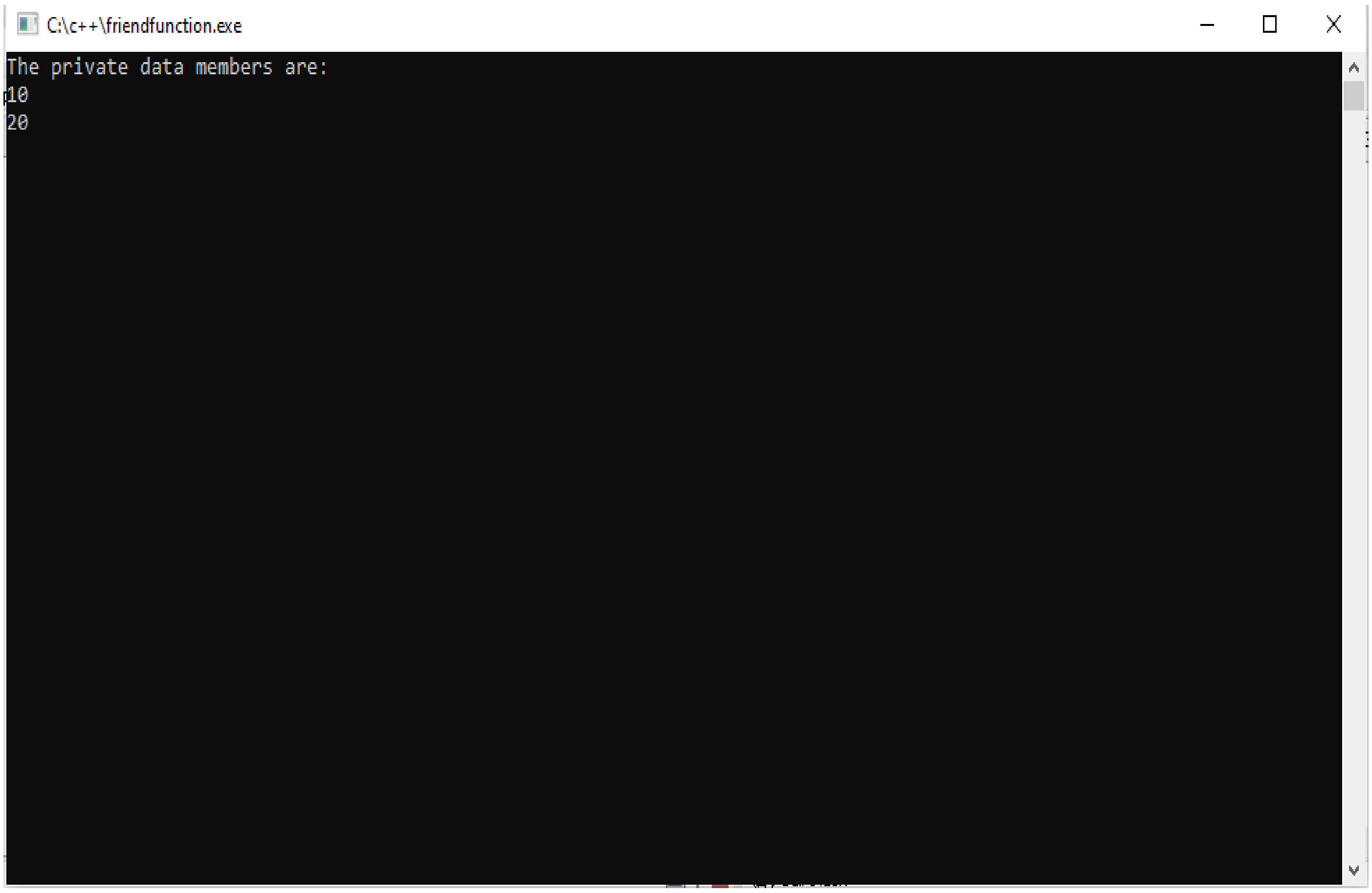


```
Enter the number
6
The factorial of a given numebr is720
```

Output of factorial program

8) Program on friend function

```
#include<iostream.h>
#include<conio.h>
class Sample
{
        private:
        int a;
        int b;
        public:
        Sample()
        {
                a = 5;
                b = 15;
        }
        friend void function1(Sample S);
};
void function1(Sample S)
{
        cout<<"The private data members are:"<<endl;
        cout<<S.a<<endl;
        cout<<S.b<<endl;
}
int main()
{
        Sample S;
        function1(S);
    getch();
    return 0;
}
```

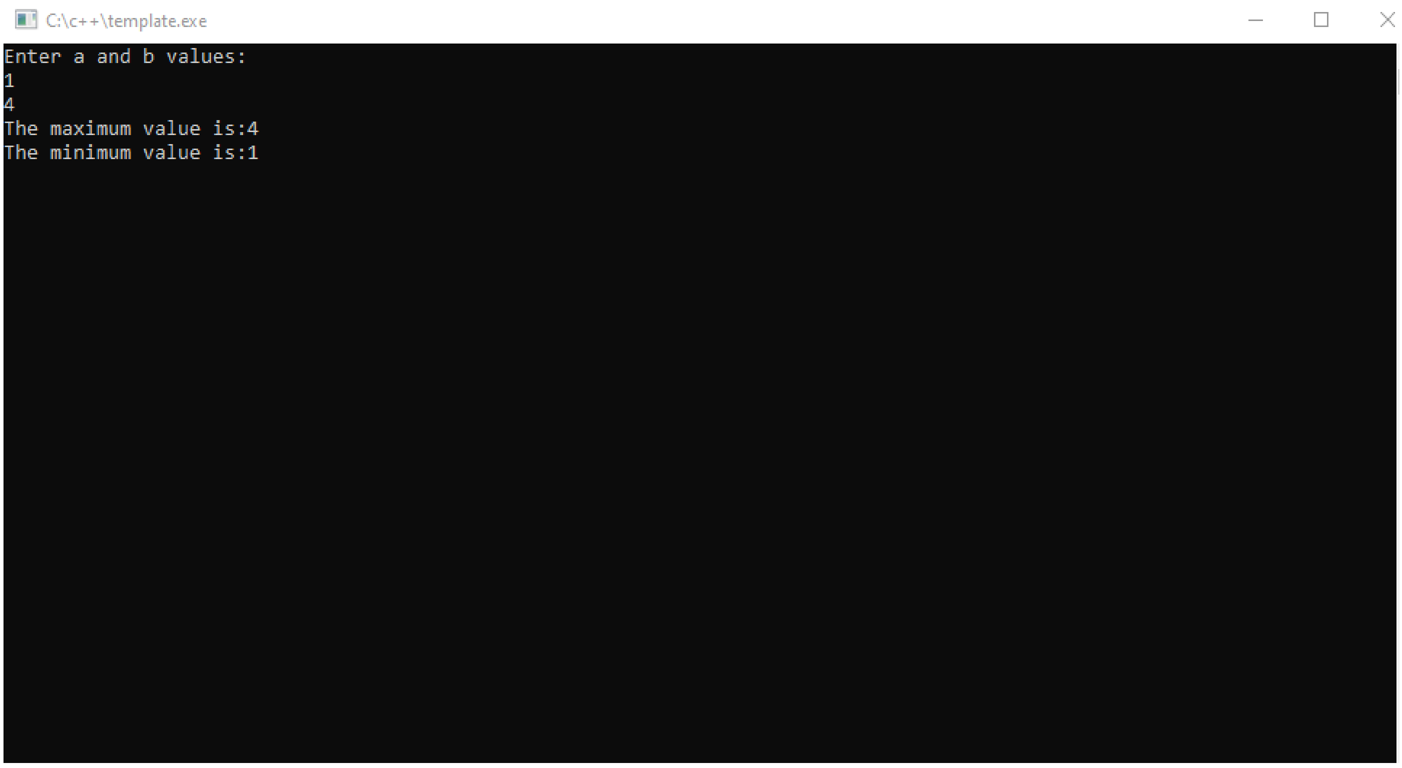


```
The private data members are:
10
20
```

Output of friend function program

9) Program on Function template

```
#include<iostream.h>
#include<conio.h>
template<class T>
T max(T a, T b)
{
        if (a>b)
        return a;
        else
        return b;
}
template<class F>
F min(F a, F b)
{
        if (a<b)
        return a;
        else
        return b;
}
int main()
{
        int a,b;
        cout<<"Enter a and b values:"<<endl;
        cin>>a>>b;
        cout<<"The maximum value is:"<<max(a,b)<<endl;
        cout<<"The minimum value is:"<<min(a,b)<<endl;
        getch();
        return 0;
}
```



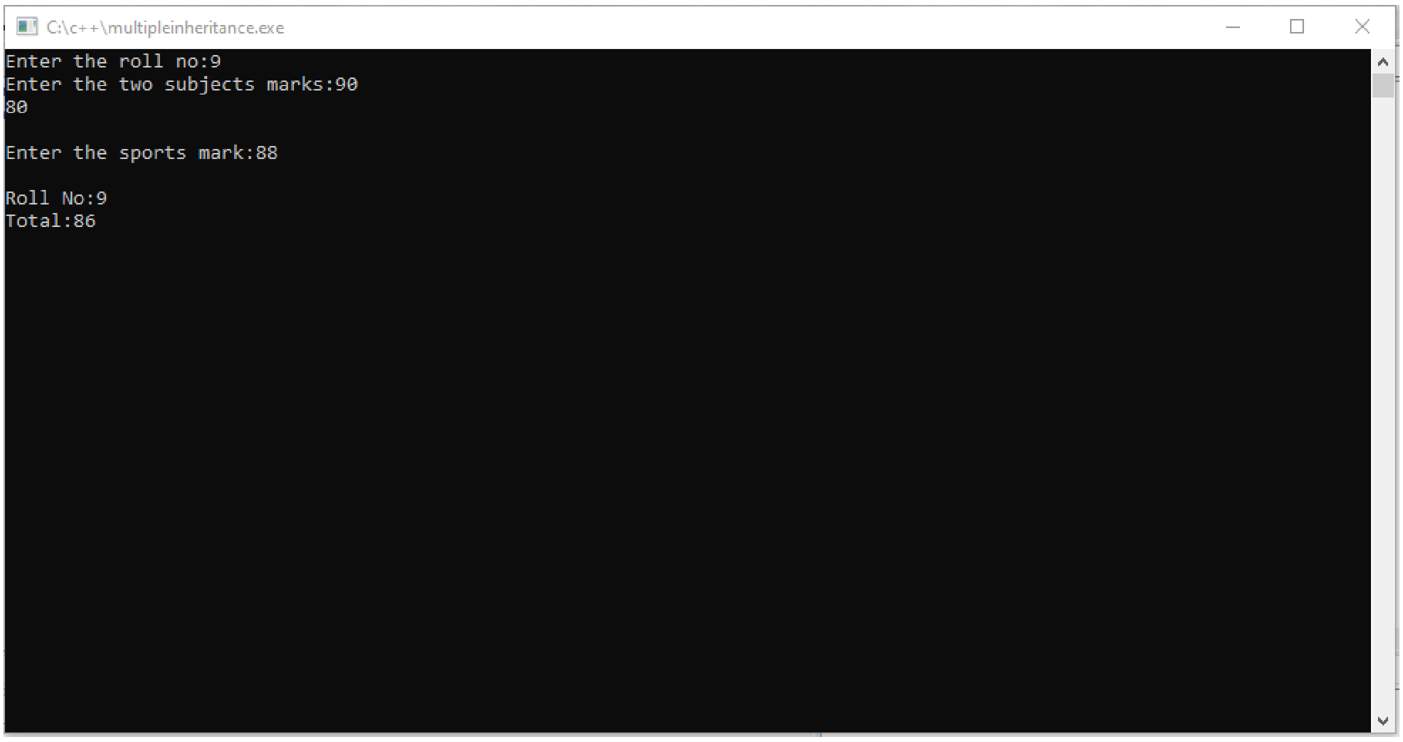
```
Enter a and b values:
1
4
The maximum value is:4
The minimum value is:1
```

Output of template program

10) <u>Program on multiple inheritances</u>

```
#include<iostream.h>
#include<conio.h>
class Student
{
        protected:
        int rno, m1, m2;
        public:
        void getdata()
        {
                cout<<"Enter the roll no:";
                cin>>rno;
                cout<<"Enter the two subjects marks:";
        cin>>m1>>m2;
    }
};
class sports
{
    protected:
    int sm;
    public:
    void getsm()
    {
        cout<<"\nEnter the sports mark:";
        cin>>sm;
    }
};
class statement: public Student, public sports
{
    int tot,avg;
    public:
    void display()
    {
        tot = (m1+m2+sm);
        avg = tot/3;
        cout<<"\nRoll No:"<<rno;
        cout<<"\nTotal:"<<avg;
    }
};
int main()
```

```
{
        statement S;
        S.getdata();
        S.getsm();
        S.display();
        getch();
        return 0;
}
```

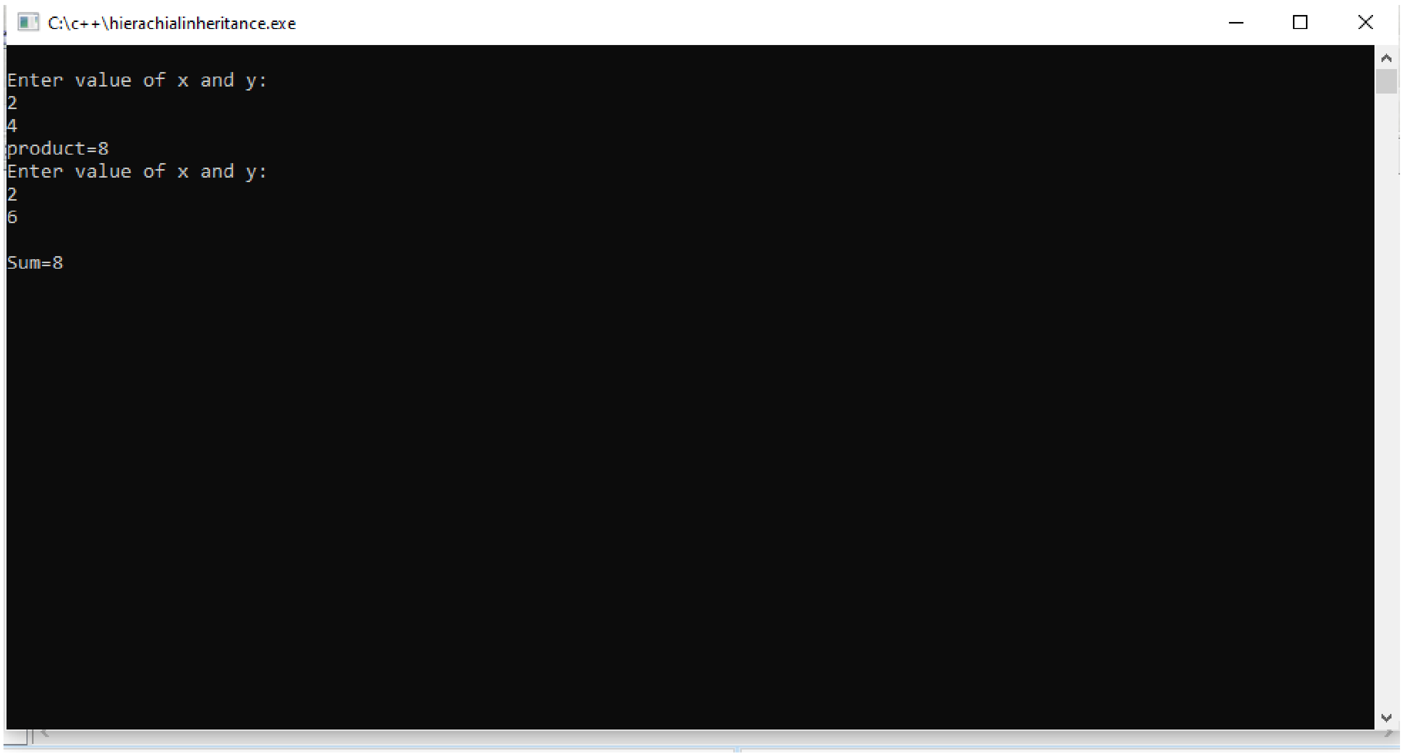


```
Enter the roll no:9
Enter the two subjects marks:90
80

Enter the sports mark:88

Roll No:9
Total:86
```

Output of multiple inheritances

11) <u>Program on hierarchical inheritance</u>

```
#include<iostream.h>
#include<conio.h>
class A             //single base class
{
        public:
        int x,y;
        void getdata()
        {
                cout<<"\nEnter value of x and y:\n";
                cin>>x>>y;
        }
};
class B: public A   //B is derieved from class base
{
        public:
        void product()
        {
                cout<<"\product="<<x*y;
        }
};
class C: public A   //C is also derieved from class base
{
        public:
        void sum()
        {
                cout<<"\nSum="<<x+y;
        }
};
int main()
{
        B obj1;       //object of derieved class B
        C obj2;       //object of derieved class C
        obj1.getdata();
        obj1.product();
        obj2.getdata();
        obj2.sum();
        getch();
        return 0;
}
```

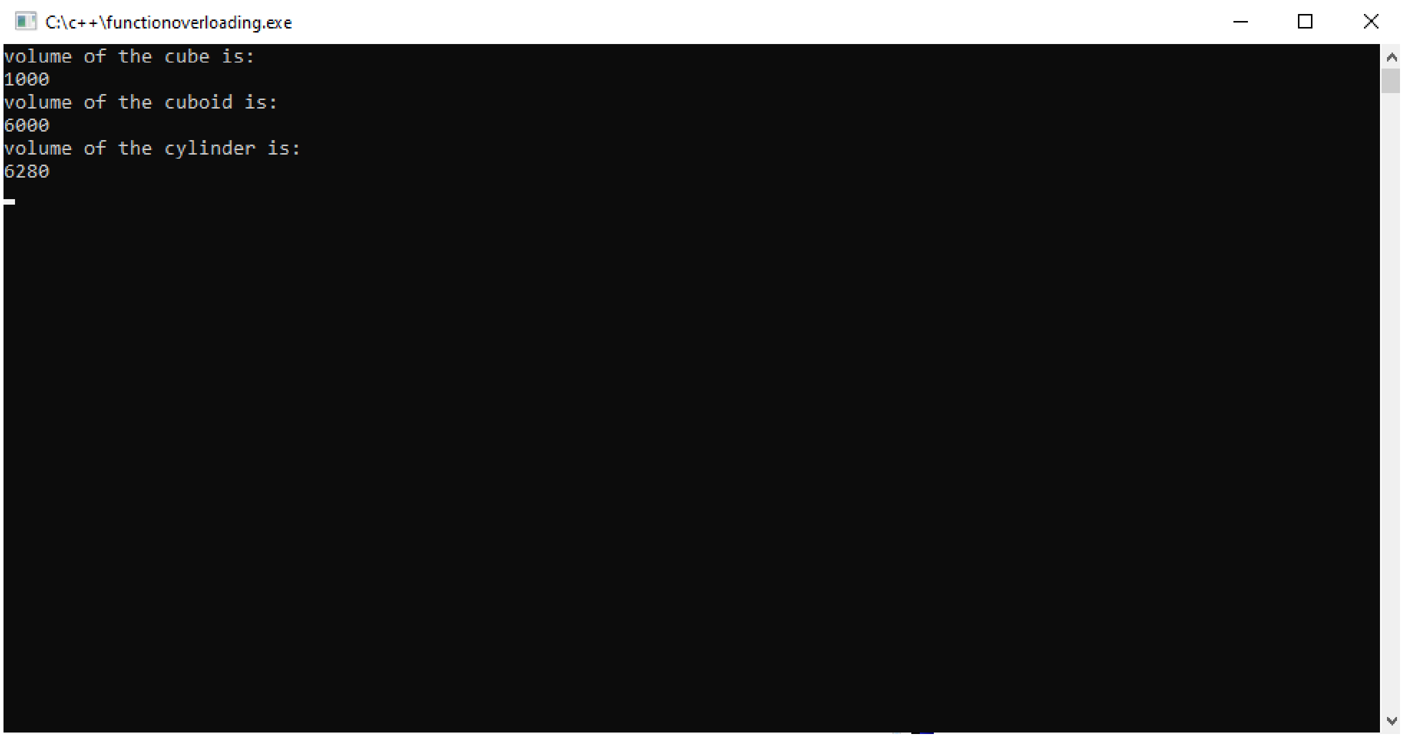


```
C:\c++\hierachialinheritance.exe

Enter value of x and y:
2
4
product=8
Enter value of x and y:
2
6

Sum=8
```

Output of hierarchical inheritance

12) <u>Program on function overloading</u>

```
#include<iostream.h>
#include<conio.h>
class calculatingvolume
{
        public:
        int volume(int a)     //for cube
        {
                return a*a*a;
        }
        int volume(int a, int b, int c)    //for cubiod
        {
                return a*b*c;
        }
        int volume (int r, int h)      //for cylinder
        {
                return 3.14*r*r*h;
        }
};
int main()
{
        calculatingvolume c;
        cout<<"volume of the cube is:"<<endl;
        cout<<c.volume(10)<<endl;
        cout<<"volume of the cuboid is:"<<endl;
        cout<<c.volume(5,10,15)<<endl;
        cout<<"volume of the cylinder is:"<<endl;
        cout<<c.volume(10,15)<<endl;
        getch();
        return 0;
}
```



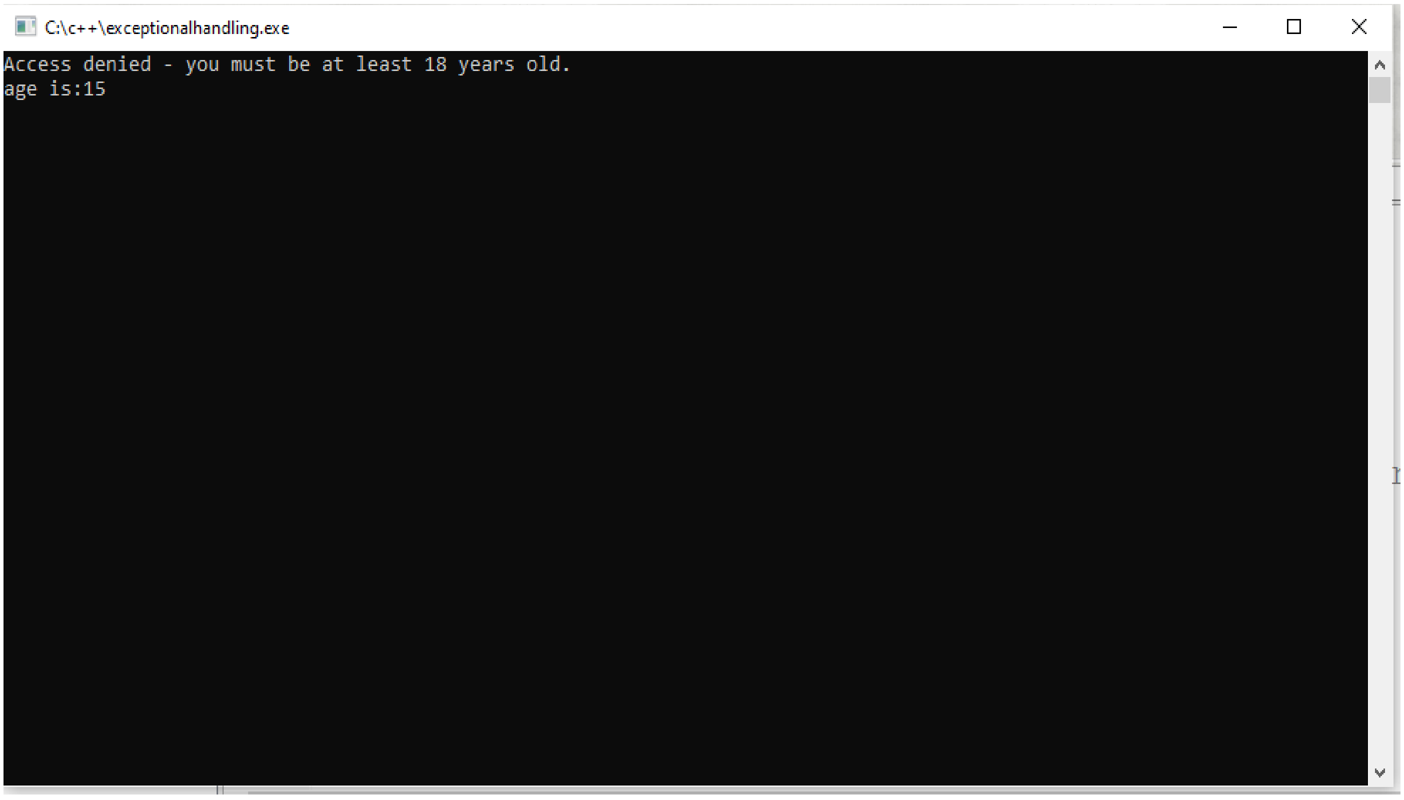
C:\c++\functionoverloading.exe

```
volume of the cube is:
1000
volume of the cuboid is:
6000
volume of the cylinder is:
6280
```

Output of function overloading program

13) <u>Program on exceptional handling</u>

```
#include<iostream.h>
#include<conio.h>
int main()
{
        try
{
        int age = 15;
        if (age> 18)
        {
                cout<<"Access granted - you are old enough.";
        }
        else
        {
                throw (age);
        }
}
catch (int myNum)
{
        cout<<"Access denied - you must be more than 18 years old.\n";
        cout<<"age is:"<<myNum;
}
  getch();
  return 0;
}
```



```
Access denied - you must be at least 18 years old.
age is:15
```

Output of exceptional handling

14) Program on Class Templates

```
    include <iostream>
template<class T1, class T2>
 class  A
{
 T1 a;
```

```
   T2 b;
 public:
      A(T1 x,T2 y)
     {
               a = x;
                  b = y;
  }
 void display()
       {
                 cout << "Values of a and b are : " << a<<" ,"<<b<<endl;
        }
             };

     int main ()
    {
            A<int,float> d(5,6.5);
         d.display();
          return 0;
            }
```

15) Program on Virtual Functions

```
include<iostream.h>
Class   base                //base class
{
Public :
Void  display()             //display() is normal member function
{
                  Cout<<"\n display base";
}
Virtual   void  show()          //show() is virtual member function
       {
       Cout<<"\n show base";
       }
 };
 Class  derived : public  base       //derived class
 {
 Public :
       Void  display()
       {
       Cout<<"\n display derived";
       }
Void  show()
{
                        Cout<<"\n show derived";
                        }
                    };
void   main()
{
        Base  b;            //base class object
        Derived  d;         //derived class object
        Base    *bptr;              //base pointer for base class
        Cout<<"\n  bptr points to base \n";
        bptr =&b;                   //storing base class object in to bptr
        bptr ->display();
        bptr ->show();
        Cout<<"\n \n  bptr points to derived \n ";
        bptr=&d;            //storing derived class object in to bptr
        bptr -> display();
        bptr -> show();
                    }
```

Output:

```
        bptr points to base
        Display base
        Show base
        bptr points to derived
        Display derived
        Show base
```