INDEX

Experiment-1

Introduction program.

```
#include  <iostream>
using namespace std;
int main()
{
 cout<<"Hello World";
 return 0;
}
```

Output

Experiment – 2

Write a program to enter mark of 6 different subjects and find out the total mark (Using cin and cout statement)

```
#include  <iostream>
using namespace std;
int main()
{
 int sub1,sub2,sub3,sub4,sub5,sub6,total;
cout<<"enter marks of sub1:";
   cin>>sub1;
cout<<"enter marks of sub2:";
cin>>sub2;
cout<<"enter marks of sub3:";
cin>>sub3;
cout<<"enter marks of sub4:";
cin>>sub4;
cout<<"enter marks of sub5:";
 cin>>sub5;
cout<<"enter marks of sub6:";
cin>>sub6;
total=sub1+sub2+sub3+sub4+sub5+sub6;
cout<<"total marks:"<<total;
return 0;
}
```

Output

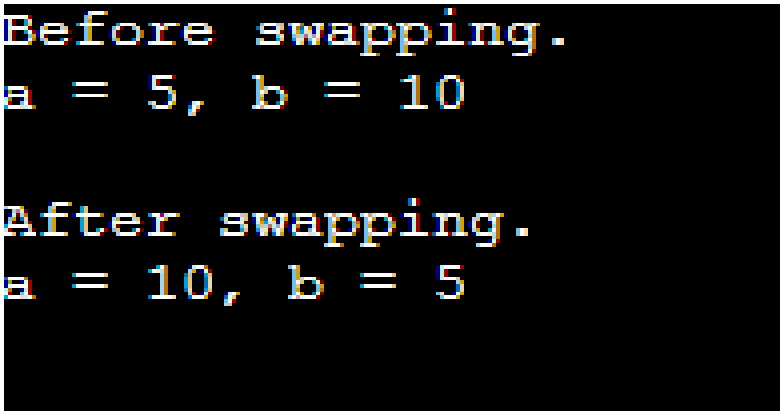
```
enter marks of sub1:20
enter marks of sub2:25
enter marks of sub3:45
enter marks of sub4:50
enter marks of sub5:67
enter marks of sub6:80
total marks:287

...Program finished with exit code 0
Press ENTER to exit console.
```

Experiment – 3

Write a function using reference variables as arguments to swap the values of pair of integers.

```
#include  <iostream>
using namespace std;
int main()
{
   int a = 5, b = 10, temp;
 cout << "Before swapping." << endl;
cout << "a = " << a << ", b = " << b << endl;
temp = a;
   a = b;
   b = temp;
cout << "\nAfter swapping." << endl;
cout << "a = " << a << ", b = " << b << endl;
return 0;
}
```

Output:

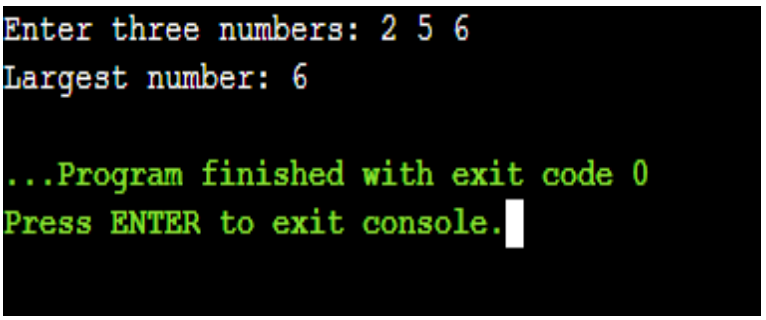


Experiment –4

Write a function to find largest of three numbers.

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

int main() {

   double n1, n2, n3;

   cout << "Enter three numbers: ";
   cin >> n1 >> n2 >> n3;

   // check if n1 is the largest number
   if(n1 >= n2 && n1 >= n3)
      cout << "Largest number: " << n1;
   else if(n2 >= n1 && n2 >= n3)
      cout << "Largest number: " << n2;

   else
      cout << "Largest number: " << n3;

   return 0;
}
```

Output

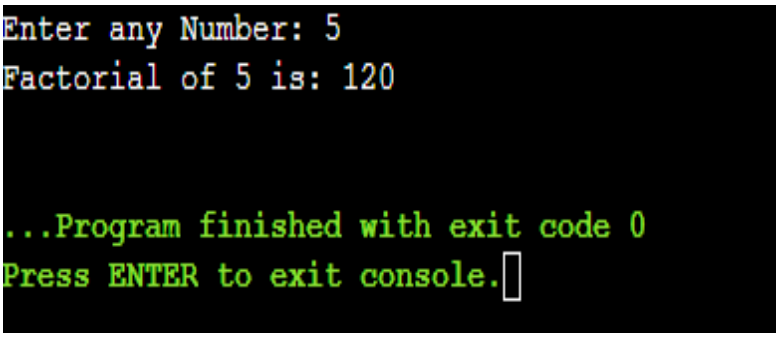


Experiment – 5

Write a program to find the factorial of a number.
(using for loop)

```
#include  <iostream>
using namespace std;
int main()
{
   int i,fact=1,number;
 cout<<"Enter any Number: ";
 cin>>number;
 for(i=1;i<=number;i++){
      fact=fact*i;
  }
  cout<<"Factorial of " <<number<<" is: "<<fact<<endl;
  return 0;
}
```

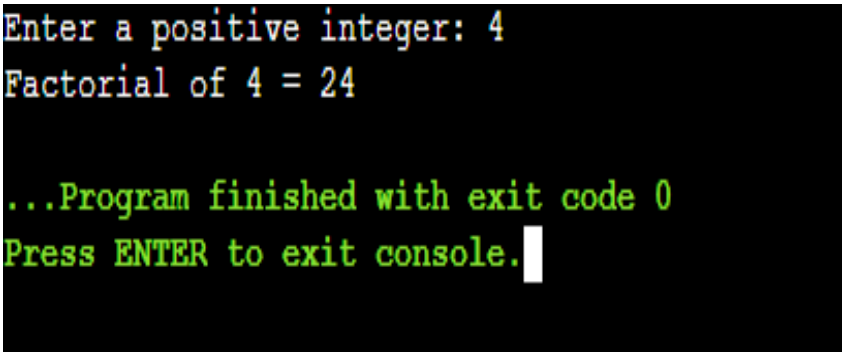
Output:-



(Using recursion)

```
#include<iostream>
using namespace std;
int factorial(int n);
int main() {
 int n;
 cout << "Enter a positive integer: ";
  cin >> n;
 cout << "Factorial of " << n << " = " << factorial(n);
 return 0;
}
int factorial(int n) {
  if(n > 1)
    return n * factorial(n - 1);
  else
    return 1;
}
```

Output



Experiment – 6

Define a class to represent a bank account which includes the following members as Data members:

a) Name of the depositor b) Account Number c) Withdrawal amount d) Balance amount in the account Member Functions: a) To assign initial values b)To deposit an amount c) To withdraw an amount after checking the balance d) To display name and balance.

```
#include<iostream>
#include<stdio.h>
#include<string.h>
using namespace std;
class bank
{
      int acno;
      char nm[100], acctype[100];
      float bal;
  public:
      bank(int acc_no, char *name, char *acc_type, float balance)
      {
            acno=acc_no;
            strcpy(nm, name);
            strcpy(acctype, acc_type);
            bal=balance;
      }
      void deposit();
      void withdraw();
      void display();
};
void bank::deposit()
{
      int damt1;
      cout<<"\n Enter Deposit Amount = ";
      cin>>damt1;
      bal+=damt1;
}
void bank::withdraw()
      int wamt1;
      cout<<"\n Enter Withdraw Amount = ";
```

```
    cin>>wamt1;
      if(wamt1>bal)
            cout<<"\n Cannot Withdraw Amount";
      bal-=wamt1;
}
void bank::display()
{
      cout<<"\n ----------------";
      cout<<"\n Accout No. : "<<acno;
      cout<<"\n Name : "<<nm;
      cout<<"\n Account Type : "<<acctype;
      cout<<"\n Balance : "<<bal;
}
int main()
{
      int acc_no;
      char name[100], acc_type[100];
      float balance;
      cout<<"\n Enter Details: \n";
      cout<<"----------------";
      cout<<"\n Accout No. ";
      cin>>acc_no;
      cout<<"\n Name : ";
      cin>>name;
      cout<<"\n Account Type : ";
      cin>>acc_type;
      cout<<"\n Balance : ";
      cin>>balance;

      bank b1(acc_no, name, acc_type, balance);
      b1.deposit();
      b1.withdraw();
      b1.display();
      return 0;
}
```

Output

```
Enter Details:
----------------------
Accout No. 254163

Name : kartik

Account Type : saving

Balance : 5000

Enter Deposit Amount = 30000

Enter Withdraw Amount = 2500

----------------------
Accout No. : 254163
Name : kartik
Account Type : saving
Balance : 32500

...Program finished with exit code 0
Press ENTER to exit console.
```

Experiment – 7

Write the above program for handling n number of account holders using array of objects.

```
#include  <iostream>
using namespace std;
int main()
{
   int n;
   cin>>n;


   int arr[n];


   int i = 0;
   while(n--){
      cin>>arr[i];
      i++;
   }
    i--;
   for(; i>=0; i--){
      cout << arr[i] << " ";
   }
  return 0;
}
```

Output

```
file handling


...Program finished with exit code 0
Press ENTER to exit console.
```

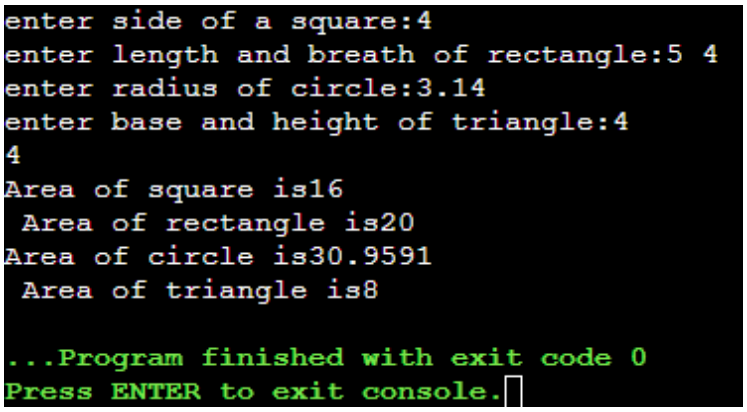
Experiment – 8

Write a C++ program to compute area of right angle triangle, equilateral triangle, isosceles triangle using function overloading concept.

```
#include<iostream>
using namespace std;
int area(int);
int area(int,int);
float area(float);
float area(float,float);
int main()
{
   int s,l,b;
   float r,bs,ht;
   cout<<"enter side of a square:";
   cin>>s;
    cout<<"enter length and breath of rectangle:";

   cin>>l>>b;
    cout<<"enter radius of circle:";
   cin>>r;
    cout<<"enter base and height of triangle:";
   cin>>bs>>ht;
   cout<<"Area of square is"<<area(s);
   cout<<"\n Area of rectangle is"<<area(l,b);
   cout<<"\nArea of circle is"<<area(r);
   cout<<"\n Area of triangle is"<<area(bs,ht);

}
int area(int s)
{
   return(s*s);
}
int area(int l,int b)
{
   return(l*b);
   }
   float area (float r)
   {
```

```
  return(3.14*r*r);
   }
   float area(float bs,float ht)
{
   return ((bs*ht)/2);
}
```

Output



Experiment – 9

Consider a publishing company that markets both book and audio cassette version to its works. Create a class Publication that stores the title (a string) and price (type float) of a publication. Derive the following two classes from the above Publication class: Book which adds a page count (int) and Tape which adds a playing time in minutes (float). Each class should have get data () function to get its data from the user at the keyboard. Write the main() function to test the Book and Tape classes by creating instances of them asking the user to fill in data with get data() and then displaying it using put data().

```
#include<iostream>
#include<string>
using namespace std;
class publication
{
   protected:
   string title;
   float price;
   public:
   publication()
   {
      title=" ";
      price=0.0;

      }
      publication(string t,float p)
      {
       title=t;
       price=p;
       }


  public:

void getdata()
{
 cout<<"Enter title of publication: ";
 cin>>title;
```

```
cout<<"Enter price of publication: ";
cin>>price;
 }

void putdata(void)
{
  cout<<"Publication titles :"<<title<<endl;
  cout<<"Publication price :"<<price<<endl;
}

};
   class book : public publication
   {
      int pagecount;
      public:
      book()
      {
         pagecount=0;
      }
      //After : base class constructor is called
      book(string t,float p,int pc):publication(t,p)
      {
         pagecount=pc;

      }
      void getdata(void)
      {
      publication::getdata();//call publication class function to get getdata
      cout <<"Enter Book Page Count :"; //Acquire book data from user
      cin>> pagecount;
      }

      void putdata(void)
      {
      publication::putdata(); //Show Publication data
      cout<< "Book page count:"<<pagecount <<endl; // Show book data
      }
   };
```

```
class CD: public publication
 {
    float time1;
    public:
    CD()
    {
       time1=0.0;
    }
    //After : base class constructor is called
    CD(string t, float p, float tim):publication(t,p)
    {
       time1=tim;
    }

    void getdata(void)
    {
    publication::getdata();
    cout <<"Enter tape's playing time:";
    cin>> time1;
    }

    void putdata(void)
    {
     publication::putdata();
     cout<<" Tape's playing time :"<< time1<<endl;
    }
 };
int main()
{
   cout<<endl<<"Book data"<<endl;
   book b("C++",230,300);
   b.putdata();
   cout<<endl<<"CD Data"<<endl;
   CD c("C++",100,120.5);
   c.putdata();

   cout<<"\n Enter New Details Of Book :\n";
   b.getdata();
   c.getdata();
```

```
    cout<<"\n\n Book data entered by user:\n";
    b.putdata();
    c.putdata();
    return 0;
}
```

Output

```
Book data
Publication titles :C++
Publication price :230
Book page count:300

CD Data
Publication titles :C++
Publication price :100
 Tape's playing time :120.5

 Enter New Details Of Book :
Enter title of publication: kaali ka karishma
Enter price of publication: Enter Book Page Count :Enter title of publication: Enter price of publication: Enter tape's playing time:

 Book data entered by user:
Publication titles :kaali
Publication price :0
Book page count:300
Publication titles :C++
Publication price :100
 Tape's playing time :120.5


...Program finished with exit code 0
Press ENTER to exit console.
```

Experiment – 10

Consider an example of declaring the examination result. Design three classes' student, exam and result. The student has data members such as roll no, name. Create the lass exam by inheriting the student class. The exam class adds data members representing the marks scored in 5 subjects. Derive the result from exam-class and it has own data members like total, avg.

```
#include<iostream>
using namespace std;
class STUDENT{
        private:
        int roll_no;
        string name;
        public:
        STUDENT(){

        }
        STUDENT(int roll, string n){
        roll_no = roll;
        name = n;
        }
};


class EXAM: public STUDENT{
        private:
        int * arr;
        int number_of_subjects;
        public:
        EXAM(){
        number_of_subjects  = 6;
        arr = new int[number_of_subjects];
        }
        int getMarks(){
                cout<<"Enter student's marks for six subject\n";
                for(int i=0; i<6; i++){
                        cout<<"Mark:    "<<(i+1)<<endl;
                        cin>>arr[i];
                }
                int sum = 0;
                for(int i=0; i<6; i++){
                        sum += arr[i];
                }
                return sum;
        }
```

```
};

class RESULT: public EXAM{
                private:
                 int total_marks;
                 public:
                 int displayTotalMarks(){
                         int  p = getMarks();


                 }
};

int main(){
                RESULT t;
cout<<"The total marks is \t"<<t.getMarks();
}
```

Output

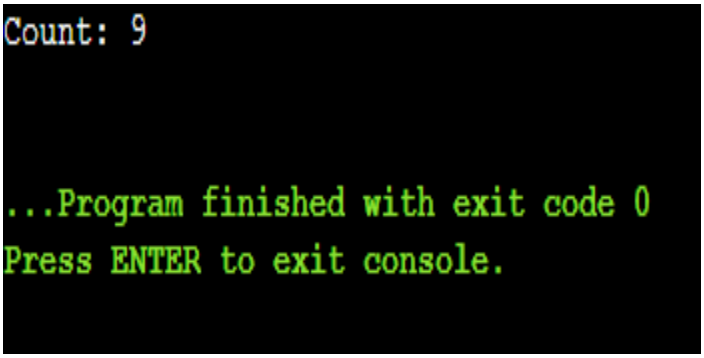
```
main.cpp: In member function 'int RESULT::displayTotalMarks()':
main.cpp:49:17: warning: no return statement in function returning non-void [-Wreturn-type]
   49 |                 }
      |                 ^
The total marks is      Enter student's marks for six subject
Mark:   1
30
Mark:   2
60
Mark:   3
80
Mark:   4
56
Mark:   5
67
Mark:   6
65
358

...Program finished with exit code 0
Press ENTER to exit console.
```

Experiment – 11

Write a program for overloading of Unary ++ operator.

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

class Count {
  private:
   int value;
public:
Count() : value(5) {}
 void operator ++ () {
       ++value;
   }
void display() {
       cout << "Count: " << value << endl;
   }
};

int main() {
   Count count1;
++count1;
count1.display();
   return 0;
}
```

Output:



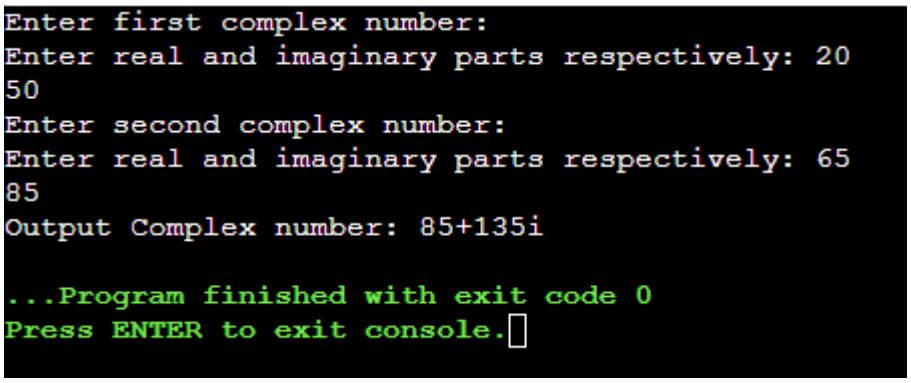
Experiment – 12

Write a program for overloading of Binary + operator.

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

class Complex {
  private:
   float real;
   float imag;

  public:

   Complex() : real(0), imag(0) {}

   void input() {
      cout << "Enter real and imaginary parts respectively: ";
      cin >> real;
      cin >> imag;
   }


   Complex operator + (const Complex& obj) {
      Complex temp;
      temp.real = real + obj.real;
      temp.imag = imag + obj.imag;
      return temp;
   }

   void output() {
      if (imag < 0)
         cout << "Output Complex number: " << real << imag << "i";
      else
         cout << "Output Complex number: " << real << "+" << imag << "i";
   }
};

int main() {
   Complex complex1, complex2, result;

   cout << "Enter first complex number:\n";
   complex1.input();

   cout << "Enter second complex number:\n";
   complex2.input();
```

```
    result = complex1 + complex2;
    result.output();

    return 0;
}
```

Output

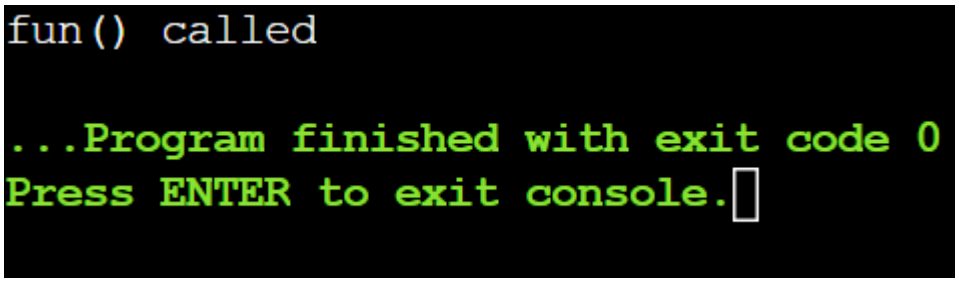


Experiment – 13

Write a program of Virtual Functions.

```
#include<iostream>
using namespace std;
class Base{
   int x;
   public:
   virtual void fun()=0;
   int getx(){ return x;}
};
 class Derived : public Base {
     int y;
     public:
     void fun(){ cout<<"fun() called";}

 };
 int main(void)
 {
     Derived d;
     d.fun();
     return 0;
 }
```

Output

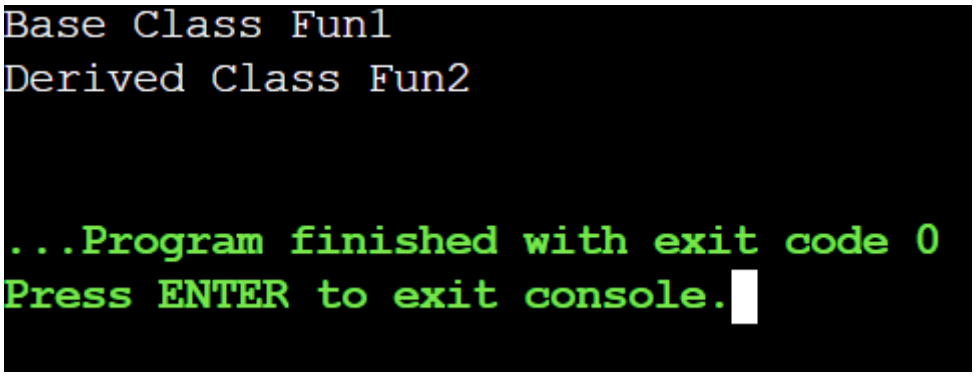


Experiment – 14

Write a program of Abstract Classes.

```
#include  <iostream>
using namespace std;
class Base
{
   public:
   void Fun1()
   {
      cout << "Base Class Fun1"<<endl;
   }
   virtual void Fun2() = 0; // Pure Virtual Function
};
class Derived:public Base
{
   public:
   void Fun2()
   {
      cout << "Derived Class Fun2"<<endl;
   }
};
int main()
{
   Derived d;
   d.Fun1();
   d.Fun2();
   return 0;
}
```

Output



Experiment – 15

Write a program to read and write from file.

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
  fstream file;
    file.open("sample.txt",ios::out);

  if(!file)
  {
     cout<<"Error in creating file!!!"<<endl;
     return 0;
  }

  cout<<"File created successfully."<<endl;

  file<<"ABCD.";

  file.close();


  file.open("sample.txt",ios::in);

  if(!file)
  {
     cout<<"Error in opening file!!!"<<endl;
     return 0;
  }

 char ch;
  cout<<"File content: ";

  while(!file.eof())
  {
     file>>ch;
     cout<<ch;
```

```
    }

    file.close();

    return 0;
}
```

Output