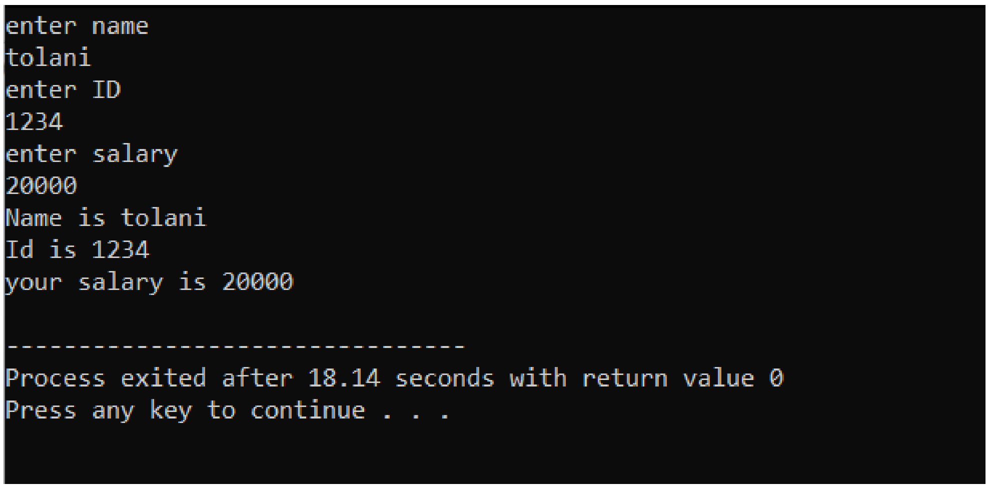
Practical 1.
Classes and Methods

(a) Design an employee class for reading and displaying the employee information, the getinfo() and displayinfo() methods will be used respectively. where getinfo() will be private method.

Program:

```
#include <iostream>
using namespace std;
class employee
{
  private:
   int emp_id;
   char emp_name[30];
   double salary;
   void getinfo()
{
   cout<<"enter name"<<endl;
   cin>>emp_name;
   cout<<"enter ID"<<endl;
   cin>>emp_id;

   cout<<"enter salary"<<endl;
   cin>>salary;
}
  public:
   void displayinfo()
{
   getinfo();
   cout<<"Name is "<<emp_name<<endl;
   cout<<"Id is "<<emp_id<<endl;
   cout<<"your salary is "<<salary<<endl;
}
};
   int main()
{
   employee e1;
   e1.displayinfo();

}
```

OUTPUT:

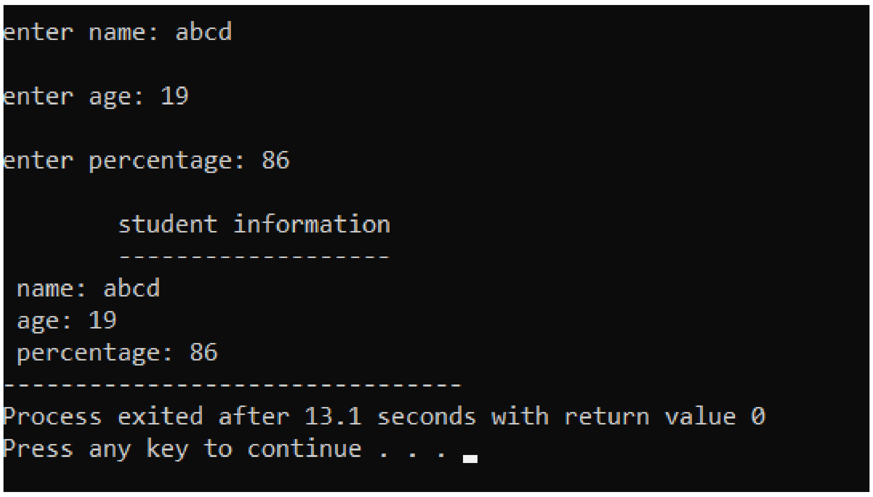


```
enter name
tolani
enter ID
1234
enter salary
20000
Name is tolani
Id is 1234
your salary is 20000

--------------------------------
Process exited after 18.14 seconds with return value 0
Press any key to continue . . .
```

(b) Design the class student containing getdata() and displaydata() as two of its methods which will be used for reading and displaying the student information respectively.where getdata() will be private method.
Program:

```
#include<iostream>
using namespace std;
class student
{
 char name[20];
 int age;
 float percentage;
 void getdata()
{
 cout<<endl<<"enter name: ";
 cin>>name;
 cout<<endl<<"enter age: ";
 cin>>age;
 cout<<endl<<"enter percentage: ";
 cin>>percentage;
}
public:
 void displaydata()
 {
 getdata();
 cout<<endl<<"\tstudent information\n";
 cout<<"\t-------------------";
 cout<<endl<<" name: "<<name;
 cout<<endl<<" age: "<<age;
 cout<<endl<<" percentage: "<<percentage;
 }
};
int main()
{
  student s;
  s.displaydata();

}
```

OUTPUT:



```
enter name: abcd

enter age: 19

enter percentage: 86

       student information
       ------------------
 name: abcd
 age: 19
 percentage: 86
----------------------------
Process exited after 13.1 seconds with return value 0
Press any key to continue . . .
```

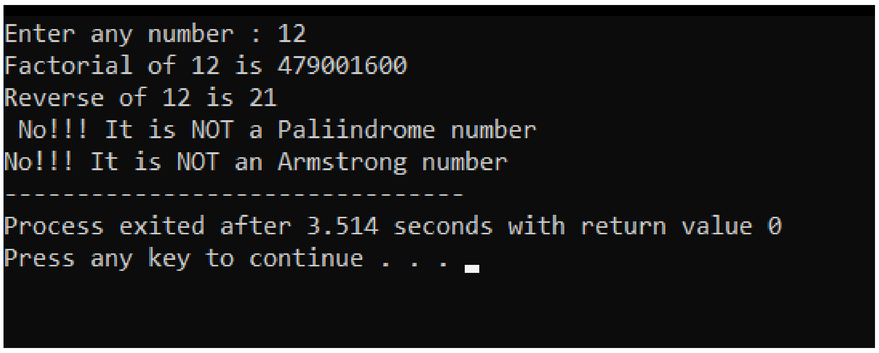
(c) Design the class demo which will contain the following methods: readno(),factorial() for calculating the factorial of a number, reverseno() will reverse the given number, ispalindrome() will check the given number is palindrome, isarmstrong() which will calculate the given number is armstrong or not.where readno() will be private method.

Program:

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

class Demo
{
   int num, len=0;

   public:
      Demo()
      {
         readNo();
      }
            void factorial()
      {
         int a = num;
         int i = 1;
         while(a>0)
         {
            i = i * a;
            a--;
            len++;
```

```
        }
        cout<<"Factorial of "<<num<<" is "<<i;
    }

    void reverse()
    {
        int a = num;
        int rev = 0;
        while(a>0)
        {
            rev = rev*10 + a%10;
            a /= 10;
        }
        cout<<"\nReverse of "<<num<<" is "<<rev;
    }

    void isPalindrome()
    {
        int a = num;
        int rev = 0;
        while(a>0)
        {
            rev =rev*10 + a%10;
            a /= 10;
        }
        if(rev == num)
        {
            cout<<"\n Yes!!! It is a Paliindrome number";
        }
        else
        {
            cout<<"\n No!!! It is NOT a Paliindrome number";
        }
    }

    void isArmstrong()
    {
        int a = num;
        int n = num;
        int rem = 0, rsl=0, i = 1;

        while(a>0)
        {
            a/=10;
            len++;
        }
        while(n>0)
        {
            rem = n%10;
            rsl += pow(rem,len);
            n /= 10;
          }
          if(rsl == num)
          {
            cout<<"\nYes!!! It is an Armstrong number";
          }
          else
          {
            cout<<"\nNo!!! It is NOT an Armstrong number";
          }
        }

    private:

        void readNo()
        {
          cout<<"Enter any number : ";
          cin>>num;
        }
};
//itvoyagers.in
int main()
{
    Demo d = Demo();
    d.factorial();
    d.reverse();
    d.isPalindrome();
    d.isArmstrong();
    return 0;
}
```

OUTPUT:

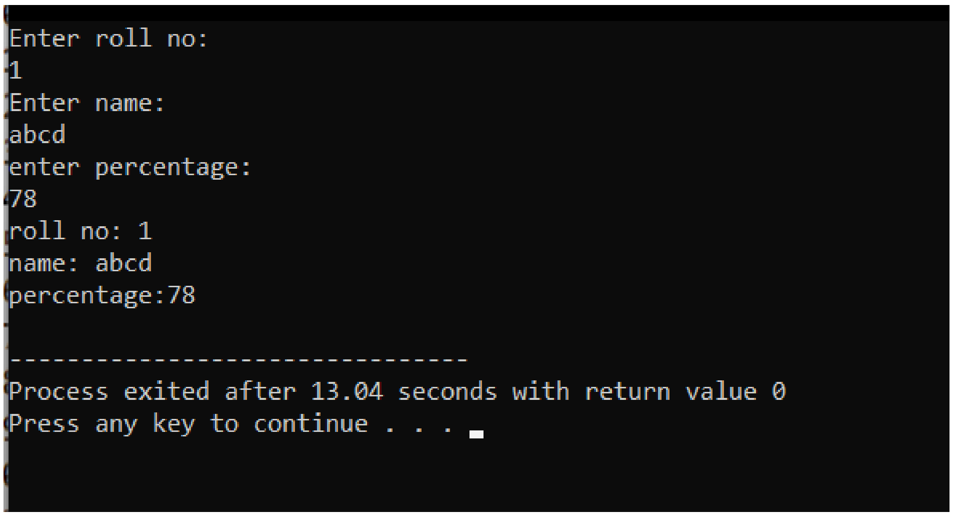


```
Enter any number : 12
Factorial of 12 is 479001600
Reverse of 12 is 21
 No!!! It is NOT a Paliindrome number
No!!! It is NOT an Armstrong number
--------------------------------
Process exited after 3.514 seconds with return value 0
Press any key to continue . . .
```

(d) Write a program to demonstrate function definition outside class and accessing class members in function definition.
Program:

```
#include<iostream>
using namespace std;
class student
{
  int roll_no;
  char name [30];
  float percentage;
    public:
      void getdata();
      void show();
};
void student:: getdata ()
{
    cout <<"Enter roll no:"<<endl;
    cin >> roll_no;
    cout << "Enter name:"<<endl;
    cin >> name;
    cout << "enter percentage:"<<endl;
    cin >> percentage;
}
void student:: show ()
{
  cout <<"roll no: "<< roll_no<<endl;
  cout << "name: "<<name<<endl;
  cout << "percentage:"<<percentage<<endl;

}
int main()
{
  student studobj;
  studobj.getdata();
  studobj.show();

  return 0;
}
```

OUTPUT:



```
Enter roll no:
1
Enter name:
abcd
enter percentage:
78
roll no: 1
name: abcd
percentage:78

--------------------------------
Process exited after 13.04 seconds with return value 0
Press any key to continue . . .
```

Practical 2.
Friend functions

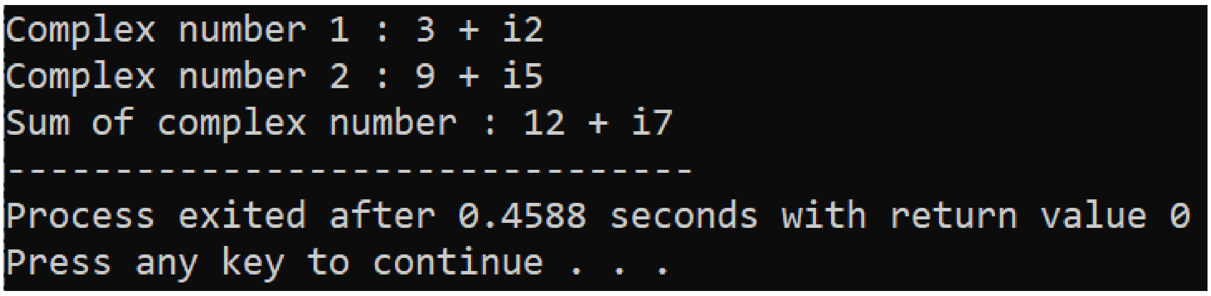
(a) Write a friend function for adding the two complex numbers, using a single class.

Program:

```
// C++ Program to Add Two Complex Numbers

// Importing all libraries
#include<iostream>
using namespace std;

// User Defined Complex class
class Complex {

        // Declaring variables
        public:
                int real, imaginary;

        // Constructor to accept
        // real and imaginary part
        Complex(int tempReal = 0, int tempImaginary = 0)
        {
                real = tempReal;
                imaginary = tempImaginary;
        }

        // Defining addComp() method
        // for adding two complex number
        Complex addComp(Complex C1, Complex C2)
        {
                // creating temporary variable
                Complex temp;

                // adding real part of complex numbers
                temp.real = C1.real + C2.real;

                // adding Imaginary part of complex numbers
                temp.imaginary = C1.imaginary + C2.imaginary;

                // returning the sum
                return temp;
        }
};

// Main Class
int main()
{
```

```
            // First Complex number
            Complex C1(3, 2);

            // printing first complex number
            cout<<"Complex number 1 : "<< C1.real
                                    << " + i"<< C1.imaginary<<endl;

            // Second Complex number
            Complex C2(9, 5);

            // printing second complex number
            cout<<"Complex number 2 : "<< C2.real
                                    << " + i"<< C2.imaginary<<endl;

            // for Storing the sum
            Complex C3;

            // calling addComp() method
            C3 = C3.addComp(C1, C2);

            // printing the sum
            cout<<"Sum of complex number : "
                                            << C3.real << " + i"
                                            << C3.imaginary;
}
```

OUTPUT:

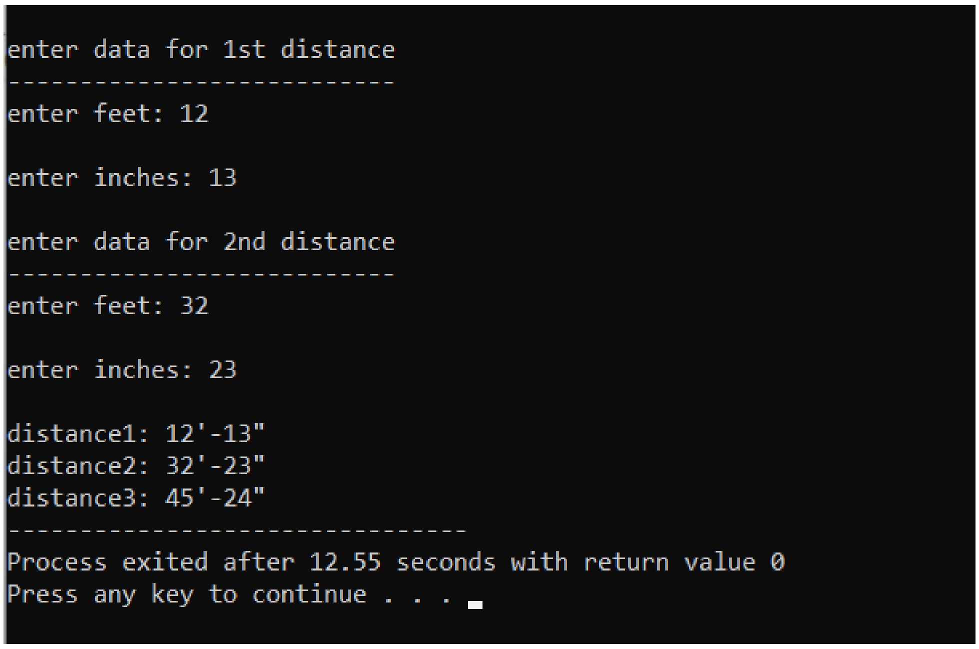


```
Complex number 1 : 3 + i2
Complex number 2 : 9 + i5
Sum of complex number : 12 + i7
--------------------------------
Process exited after 0.4588 seconds with return value 0
Press any key to continue . . .
```

(b) Write a friend function for adding the two different distances and display its sum, using two classes.
Program:

```
#include<iostream>
using namespace std;
class distance2;
class distance1
{
  int feet;
  int inch;
  public:
  void getdata()
{
  cout<<"\nenter feet: ";
  cin>>feet;
  cout<<"\nenter inches: ";
  cin>>inch;
}
  void showdata()
{
  cout<< feet <<"'-" <<inch<<"\"";
}
  friend void sum(distance1, distance2);
};
  class distance2
{
  int feet,inch;
  public:
  void getdata()
{
  cout<<"\nenter feet: ";
  cin>>feet;
  cout<<"\nenter inches: ";
  cin>>inch;
}
void showdata()
{
 cout<< feet<<"'-"<<inch <<"\"" ;
}
 friend void sum(distance1, distance2);
};
 void sum(distance1 d1, distance2 d2)
 {
  int f=d1.feet+d2.feet;
  int i=d1.inch+d2.inch;
  if(i>=12)
{
 i=i-12;
 f++;
}
 cout<< f<<"'-"<<i <<"\"" ;
 }
int main()
{
 distance1 obj1;
 distance2 obj2;
 cout<<"\nenter data for 1st distance \n";
 cout<<"---------------------------";
 obj1.getdata();
 cout<<"\nenter data for 2nd distance \n";
 cout<<"---------------------------";
 obj2.getdata();
 cout<<"\ndistance1: ";
 obj1.showdata();
 cout<<"\ndistance2: ";
 obj2.showdata();
 cout<<"\ndistance3: ";
 sum(obj1,obj2);

 return 0;
}
```

OUTPUT:



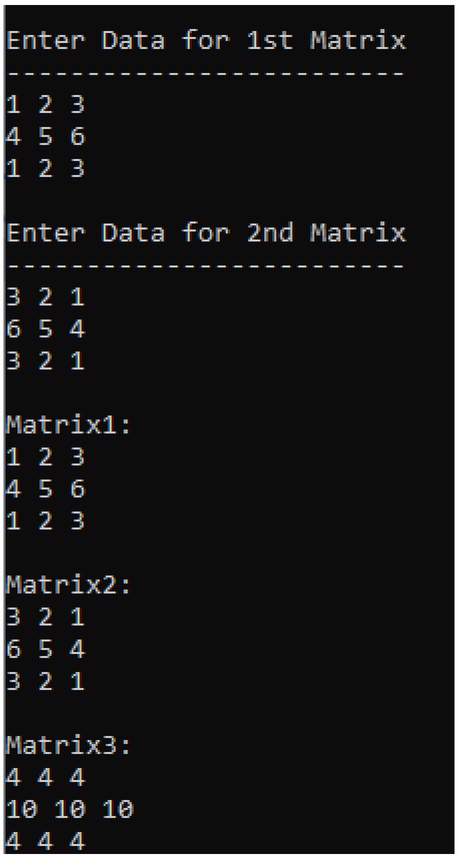
```
enter data for 1st distance
---------------------------
enter feet: 12

enter inches: 13

enter data for 2nd distance
---------------------------
enter feet: 32

enter inches: 23

distance1: 12'-13"
distance2: 32'-23"
distance3: 45'-24"
--------------------------------
Process exited after 12.55 seconds with return value 0
Press any key to continue . . .
```

(c) Write a friend function for adding the two matrix from two different classes and display its sum

Program:

```
#include<iostream>
using namespace std;
class matrix2;
class matrix1
{
int a[3][3];
public:
void getData()
{
for(int i=0;i<3;i++)
 {
 for(int j=0;j<3;j++)
 cin>>a[i][j];
 }
}
void showData()
{
for(int i=0;i<3;i++)
 {
 for(int j=0;j<3;j++)
 cout<<a[i][j]<<" ";
 cout<<endl;
 }
}
 friend void sum(matrix1, matrix2);
};
class matrix2
{
int a[3][3];
public:
void getData()
{
for(int i=0;i<3;i++)
 {
 for(int j=0;j<3;j++)
 cin>>a[i][j];
 }
}
void showData()
{
for(int i=0;i<3;i++)
 {
 for(int j=0;j<3;j++)
 cout<<a[i][j]<<" ";
 cout<<endl;
 }
}
 friend void sum(matrix1, matrix2);
};
 void sum(matrix1 m1, matrix2 m2)
 {
 int a[3][3];
 for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
 {
 a[i][j]=m1.a[i][j] + m2.a[i][j];
 cout<<a[i][j]<<" ";
 }
 cout<<endl;
}
 }
int main()
```

```
{

matrix1 obj1;
matrix2 obj2;
cout<<"\nEnter Data for 1st Matrix \n";
cout<<"-------------------------\n";
obj1.getData();
cout<<"\nEnter Data for 2nd Matrix \n";
cout<<"-------------------------\n";
obj2.getData();
cout<<"\nMatrix1: \n";
obj1.showData();
cout<<"\nMatrix2: \n";
obj2.showData();
cout<<"\nMatrix3: \n";
sum(obj1,obj2);



}
```



```
Enter Data for 1st Matrix
-------------------------
1 2 3
4 5 6
1 2 3

Enter Data for 2nd Matrix
-------------------------
3 2 1
6 5 4
3 2 1

Matrix1:
1 2 3
4 5 6
1 2 3

Matrix2:
3 2 1
6 5 4
3 2 1

Matrix3:
4 4 4
10 10 10
4 4 4
```

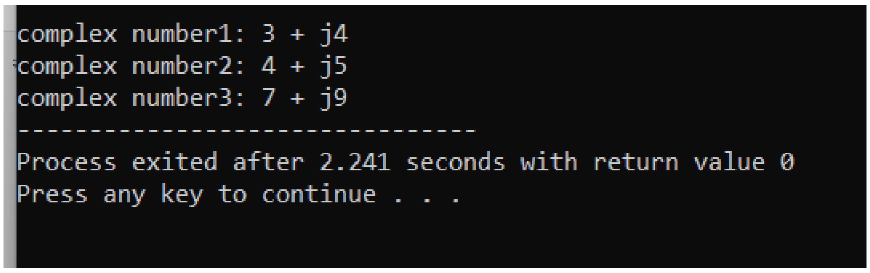
Practical 3.
Constructors and Method Overloading

(a) Design a class complex for adding the two complex numbers and also show the use of constructor.

Program:

```
#include<iostream>
using namespace std;
class complex
{
float n,m;
public:
complex()
{
 n=0;
 m=0;
}
 complex(int a, int b)
{
 n=a;
 m=b;
}
void showdata()
{
 cout<< n <<" + j"<< m ;
}
complex sum(complex c1)
{
 complex c3;
 c3.n=n+c1.n;
 c3.m=m+c1.m;
 return c3;
}
};
int main()
{
 complex obj1(3,4);
 complex obj2(4,5);
 complex obj3;
 obj3=obj1.sum(obj2);
 cout<<"\ncomplex number1: ";
 obj1.showdata();
 cout<<"\ncomplex number2: ";
 obj2.showdata();
 cout<<"\ncomplex number3: ";
 obj3.showdata();
}
```

OUTPUT:



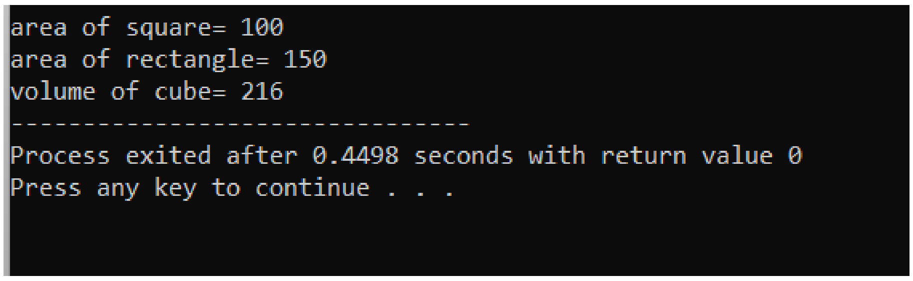
```
complex number1: 3 + j4
complex number2: 4 + j5
complex number3: 7 + j9
--------------------------------
Process exited after 2.241 seconds with return value 0
Press any key to continue . . .
```

(b) Design a class geometry containing the methods area() and volume() and also overload the area() function .

Program:

```
#include<iostream>
using namespace std;
class geometry
{
 int l,b;

 public:
 int area(int x)
{
l=b=x;
return(l*b);
}
 int area(int x, int y)
{
 l=x;
 b=y;
 return(l*b);
}
 int volume(int x)
{
 l=x;
 return(l*l*l);
}
};
 int main()
{
 geometry g;

 cout<<"\narea of square= "<<g.area(10);
 cout<<"\narea of rectangle= "<<g.area(10,15);
 cout<<"\nvolume of cube= "<<g.volume(6);

 return 0;
}
```

OUTPUT:



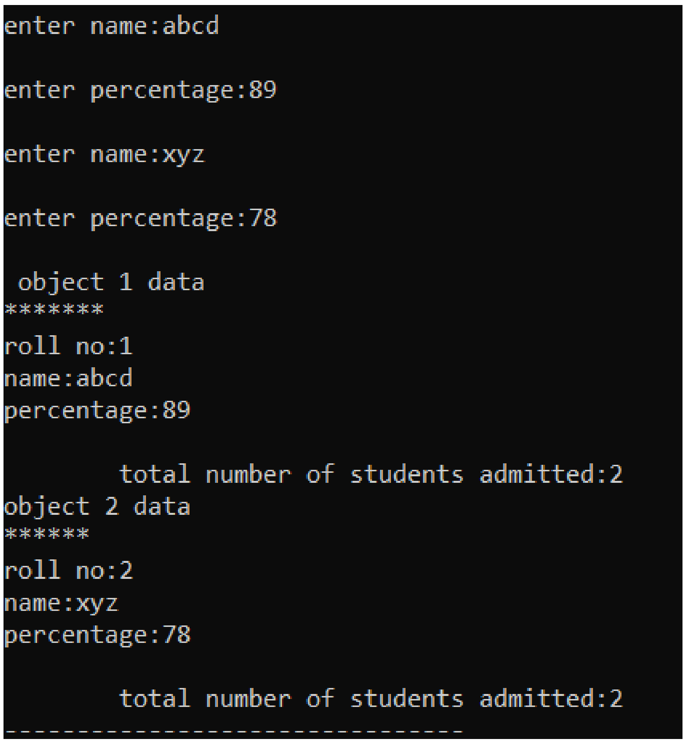
```
area of square= 100
area of rectangle= 150
volume of cube= 216
--------------------------------
Process exited after 0.4498 seconds with return value 0
Press any key to continue . . .
```

(c) Design a class static demo to show the implementation of static variable and static function.

Program:

```
#include<iostream>
#include<conio.h>
#include<stdio.h>
using namespace std;
class student

{
  int roll_no;
  char name[30];
  float percent;
  static int c;
  public:
  void get()
{
 cout<<"\nenter name:";
 cin>>name;
 cout<<"\nenter percentage:";
 cin>>percent;
 roll_no= ++c;
}
 void show()
{
 cout<<"\nroll no:"<<roll_no;
 cout<<"\nname:"<<name;
 cout<<"\npercentage:"<<percent;
 cout<<"\n\n\ttotal number of students admitted:"<<c;
}

};
int student::c;
int main()
{
 student s1,s2;
 s1.get();
 s2.get();
 cout<<"\n object 1 data";
 cout<<"\n*******";
 s1.show();
 cout<<"\nobject 2 data";
 cout<<"\n******";
 s2.show();

 return 0;
}
```

OUTPUT:



```
enter name:abcd

enter percentage:89

enter name:xyz

enter percentage:78

 object 1 data
*******
roll no:1
name:abcd
percentage:89

        total number of students admitted:2
object 2 data
******
roll no:2
name:xyz
percentage:78

        total number of students admitted:2
--------------------------------
```

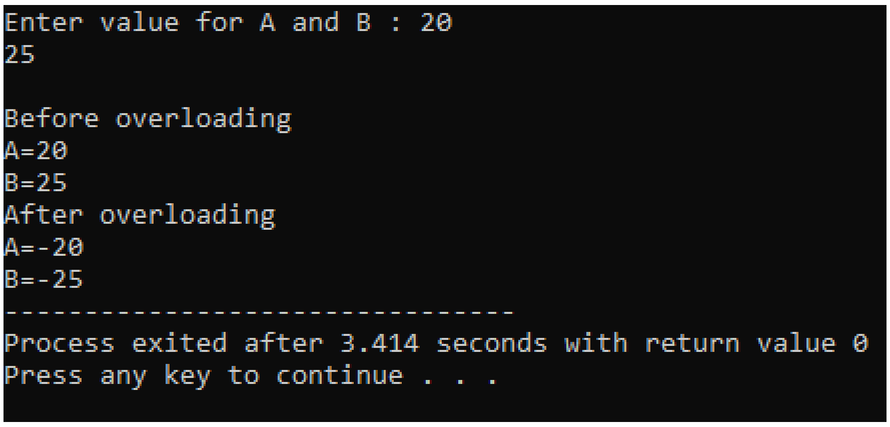
Practical 4.
Operator overloading

(a) Overload the operator unary(-) for demonstrating operator overloading.

Program:

```
#include<iostream>
using namespace std;
class Minus
{
   int a,b;
   public:
      void get()
      {
         cout<<"Enter value for A and B : ";
         cin>>a>>b;
      }
      void show()
      {
         cout<<endl<<"A="<<a<<endl<<"B="<<b;
      }
      void operator -()
      {
         a=-a;
         b=-b;
      }
};
int main()
{
   Minus s;
   s.get();
   cout<<endl<<"Before overloading";
   s.show();
   -s;
   cout<<endl<<"After overloading";
   s.show();
   return 0;
}
}
```

OUTPUT:



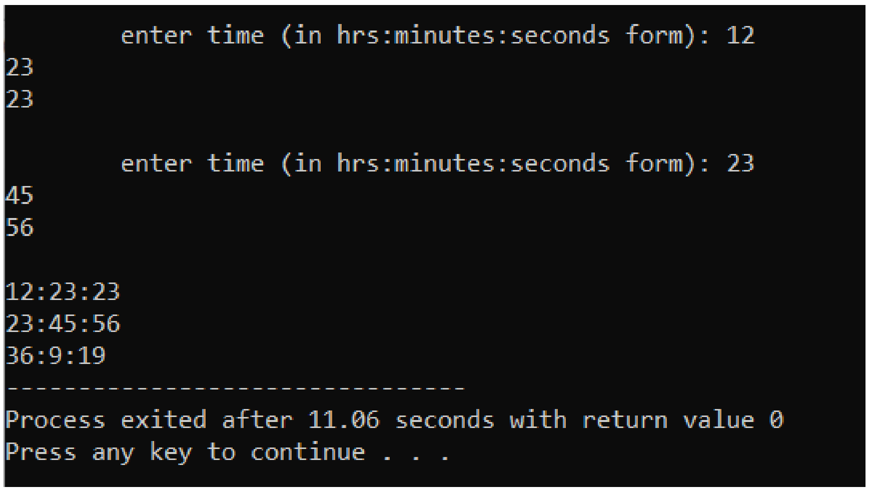
```
Enter value for A and B : 20
25

Before overloading
A=20
B=25
After overloading
A=-20
B=-25
--------------------------------
Process exited after 3.414 seconds with return value 0
Press any key to continue . . .
```

(b) Overload the operator + for adding the timings of two clocks, and also pass objects as an argument.

Program:

```
#include<iostream>
using namespace std;
class time
{
 int hrs,min,sec;
 public:
 void get()
{
 cout<<"\n\tenter time (in hrs:minutes:seconds form): ";
 cin>>hrs>>min>>sec;
}
void show()
{
 cout<<"\n"<<hrs<<":"<<min<<":"<<sec;
}
time operator +(time t2)
{
 time t3;
 t3.sec=sec + t2.sec;
 t3.min=min + t2.min + (t3.sec/60);

 t3.sec=t3.sec%60;
 t3.hrs=hrs + t2.hrs + (t3.min/60);
 t3.min=t3.min%60;
 return t3;
}
};
int main()
{
 time t1,t2,t3;
 t1.get();
 t2.get();
 t1.show();
 t2.show();
 t3=t1 + t2;
 t3.show();

 return 0;
}
```

OUTPUT:

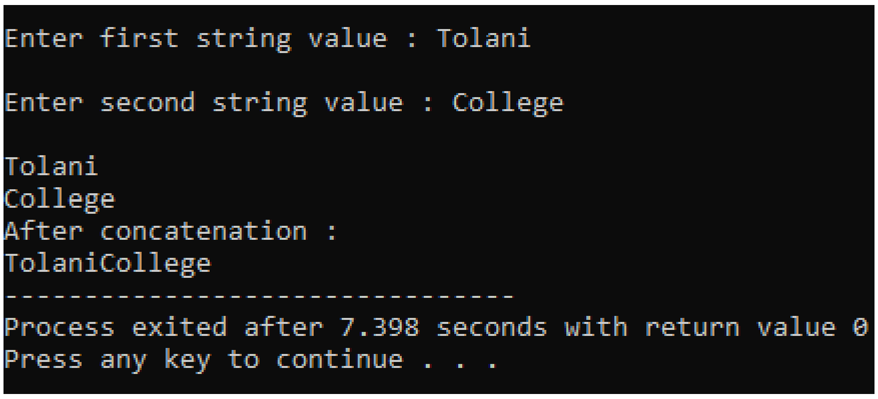


```
        enter time (in hrs:minutes:seconds form): 12
23
23

        enter time (in hrs:minutes:seconds form): 23
45
56

12:23:23
23:45:56
36:9:19
--------------------------------
Process exited after 11.06 seconds with return value 0
Press any key to continue . . .
```

(c) Overload the + for concatenating the two strings. for e.g “py” + “thon” = python.
Program: -

```
#include<iostream>
#include<string.h>
using namespace std;
class Concate
{
    char s[100];
    public:
        void get()
        {
            cin>>s;
        }
        void disp()
        {
            cout<<endl<<s;
        }
        Concate operator +(Concate s1)
        {
            Concate s2;
            strcpy(s2.s,s);
            strcat(s2.s,s1.s);
            return s2;
        }
};
int main()
{
    Concate s1,s2,s3;
    cout<<endl<<"Enter first string value : ";
    s1.get();
    cout<<endl<<"Enter second string value : ";
    s2.get();
    s1.disp();
    s2.disp();
    s3=s1+s2;
    cout<<endl<<"After concatenation : ";
    s3.disp();
    return 0;
}
```

OUTPUT:



```
Enter first string value : Tolani

Enter second string value : College

Tolani
College
After concatenation :
TolaniCollege
--------------------------------
Process exited after 7.398 seconds with return value 0
Press any key to continue . . .
```

Practical 5.
Inheritance

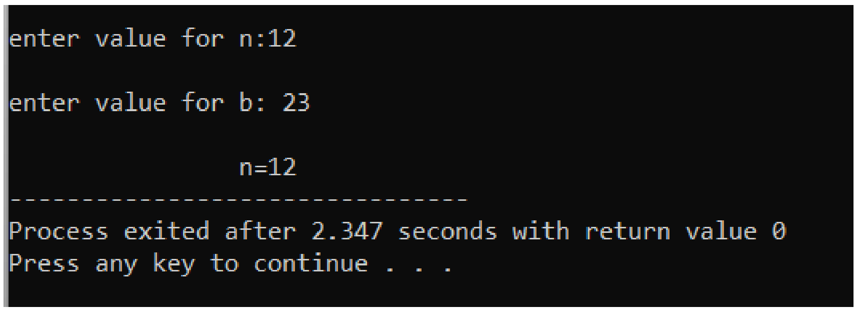
(a) Design a class for single level inheritance using public and private type derivation.

• Using public type derivation:

Program:

```
#include<iostream>
using namespace std;
class base
{
 int n;
 public:
 void get()
{
 cout<<"\nenter value for n:";
 cin>>n;
}
void show()
{
 cout<<"\n\t\tn="<<n;
}
};
class derived:public base
{
 int b;
 public:
 void get()

{
 base::get();
 cout<<"\nenter value for b: ";
 cin>>b;
}
};
int main()
{
 derived d1;
 d1.get();
 d1.show();

 return 0;
}
```

OUTPUT:



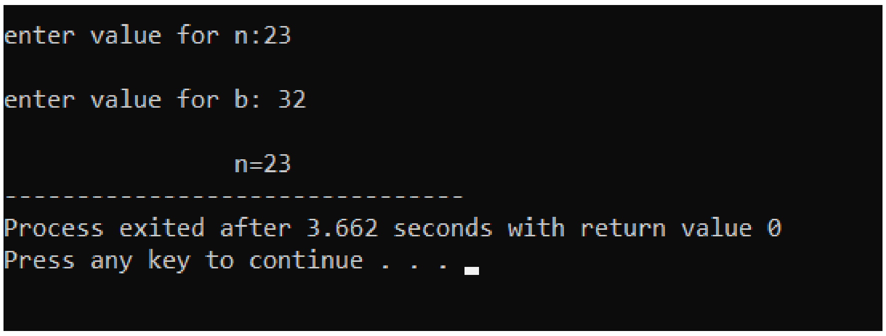
```
enter value for n:12

enter value for b: 23

                n=12
--------------------------------
Process exited after 2.347 seconds with return value 0
Press any key to continue . . .
```

• Using private type derivation:

Program:

```
#include<iostream>
using namespace std;
class base
{
 int n;
 public:
 void get()
{
 cout<<"\nenter value for n:";
 cin>>n;
}

void show()
{
 cout<<"\n\t\tn="<<n;
}
};
class derived:private base
{
 int b;
 public:
 void get()
{
 base::get();
 cout<<"\nenter value for b: ";
 cin>>b;
}
void display()
{
 show();
}
};
int main()

{
 derived d1;
 d1.get();
// d1.show(); not accessible as its scope is private
 d1.display();

 return 0;
}
```

OUTPUT:



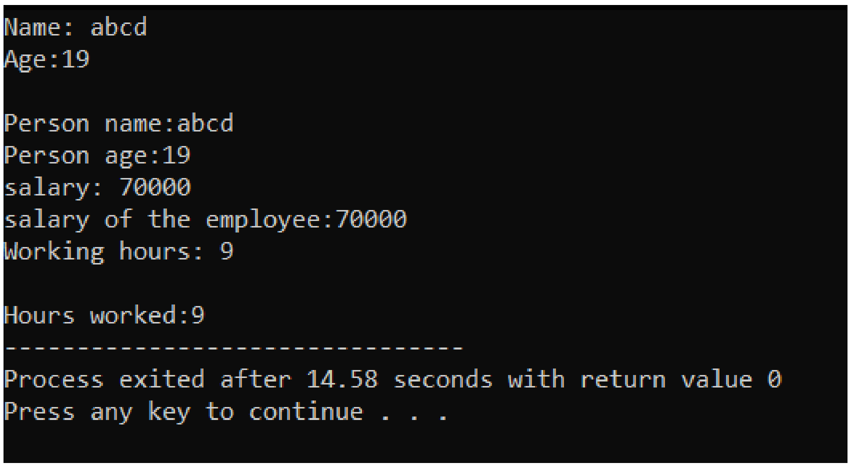
```
enter value for n:23

enter value for b: 32

                n=23
--------------------------------
Process exited after 3.662 seconds with return value 0
Press any key to continue . . .
```

(b) Design a class for multiple inheritance.

Program:

```
#include<iostream>
using namespace std;
class person
{
        protected:
                int age;
                char name[20];
   public:
        void get()
        {
                cout<<"Name: ";
                cin>>name;
                cout<<"Age:";
                cin>>age;
                        }
        void disp()
        {
                cout<<endl<<"Person name:"<<name;
                cout<<endl<<"Person age:"<<age;
                }
};
class employee
{
        protected:
                float sal;
        public:
                void esal()
                        {
                        cout<<endl<<"salary: ";
                        cin>>sal;
                        cout<<"salary of the employee:"<<sal;
                }
};
class empful:public person,public employee
{
                protected:
                float hours;
        public:
                void eworkhrs()
                {
                        cout<<endl<<"Working hours: ";
                        cin>>hours;
                }
                void printhrs()
                {
                        cout<<endl<<"Hours worked:"<<hours;
                }
};
int main()
{
        empful e;
        e.get();
        e.disp();
        e.esal();
        e.eworkhrs();
        e.printhrs();
        return 0;
}
```

OUTPUT:



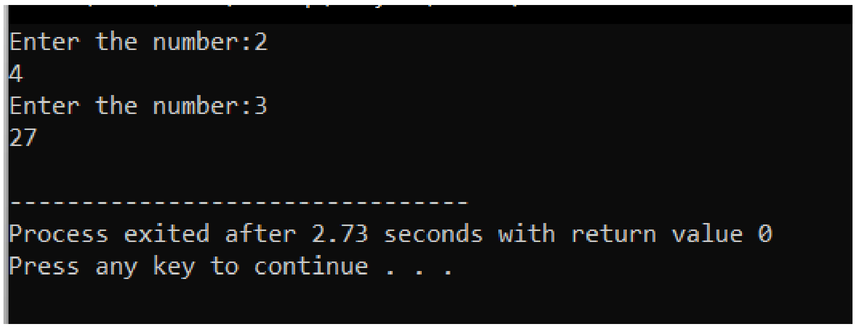
```
Name: abcd
Age:19

Person name:abcd
Person age:19
salary: 70000
salary of the employee:70000
Working hours: 9

Hours worked:9
--------------------------------
Process exited after 14.58 seconds with return value 0
Press any key to continue . . .
```

(c) Implement the hierarchical inheritance.
Program –

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

```
class number{

        public:
                int a;

                void getdata(){
                        cout<<"Enter the number:";
                        cin>>a;
                }
};

class square : public number
{
        public:
                int sqr;
                int getsquare(){
                        sqr = a*a;
//                      cout<<endl;
                        return sqr;
        }

};

class cube : public number
{
        public:
                int getcube(){
                        return a*a*a;
        }
};
int main(){
        square sqa1;
        sqa1.getdata();
        cout<<sqa1.getsquare()<<endl;
        cube cu1;
        cu1.getdata();
        cout<<cu1.getcube()<<endl;

        return 0;
}
```

OUTPUT:



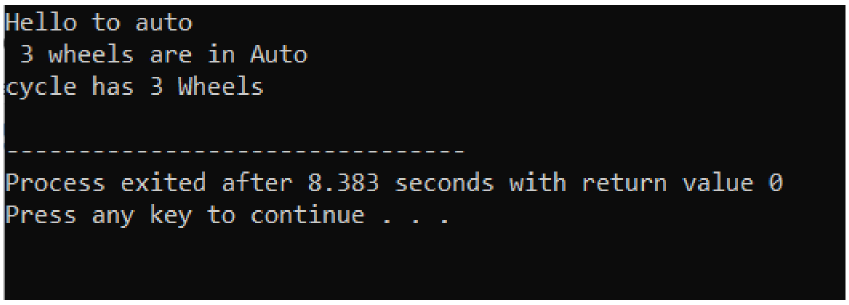
```
Enter the number:2
4
Enter the number:3
27

--------------------------------
Process exited after 2.73 seconds with return value 0
Press any key to continue . . .
```

(d) Design for Multilevel Inheritance

Program:

```
// Multilevel Inheritance in c++
#include<iostream>
using namespace std;
class Auto
{
  public:
    Auto()
    {
      cout << "Hello to auto\n";
    }
};
class threewheeler: public Auto
{  public:
    threewheeler()
    {
      cout << " 3 wheels are in Auto \n";
    }
};
class Cycle: public threewheeler {
   public:
     Cycle()
     {
       cout << "cycle has 3 Wheels\n";
     }
};
int main()
{
    Cycle obj;
    return 0;
}
```

OUTPUT:



```
Hello to auto
 3 wheels are in Auto
cycle has 3 Wheels

--------------------------------
Process exited after 8.383 seconds with return value 0
Press any key to continue . . .
```

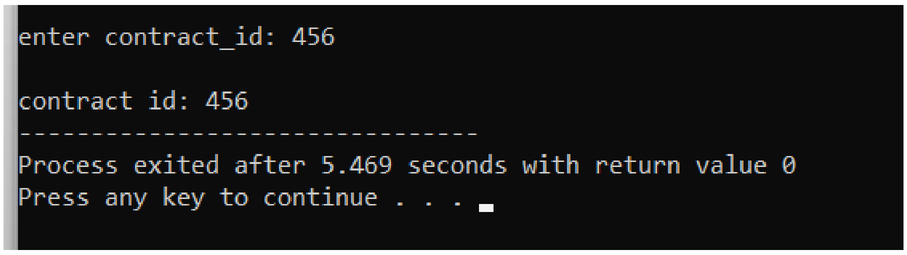
Practical 6.
Virtual functions and abstract classes

(a) Implement the concept of method overriding.

Program:

```
#include<iostream>
using namespace std;
class employee
{
int emp_code,age;
char name[30], qualification[30];
public:
void get()
{
 cout<<"\nenter employee id: ";
 cin>>emp_code;
 cout<<"\nenter employee name: ";
 cin>>name;
 cout<<"\nenter employee age: ";
 cin>>age;
 cout<<"\nenter employee qualification: ";
 cin>>qualification;
}
void show()
{
 cout<<"\n\nemployee id: "<<emp_code;
 cout<<"\tname: "<<name;
 cout<<"\nage: "<<age<<"\t\tqualification: "<<qualification;
}
};
class contract_employee: public employee
{
int contract_id;
public:
void get()
{
 cout<<"\nenter contract_id: ";
 cin>>contract_id;
}
void show()
{
 cout<<"\ncontract id: "<<contract_id;
}
};
int main()
{
 contract_employee ce;
 ce.get();
 ce.show();
 return 0;
}
```

OUTPUT:



```
enter contract_id: 456

contract id: 456
--------------------------------
Process exited after 5.469 seconds with return value 0
Press any key to continue . . .
```

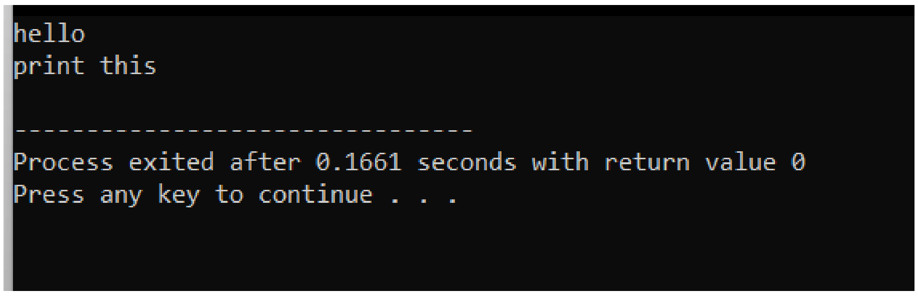
(b) Show the use of virtual function

Program:

```
//virtual function
#include<iostream>
using namespace std;
class Birds
{
        public:
        virtual void air(){
                cout<<"go and fly"<<endl;
   }
};

 class Animal: public Birds
 {
        public:
        void air() {
        cout<<"hello"<<endl;
         }
};
class Jungle: public Birds
{
        void air()
        {
                cout<<"print this"<<endl;
        }

};
void function(Birds *luck)
{luck->air(); }

int main()
{
   Birds *spo;
```

```
        Animal aniobj;
        Jungle junobj;
        spo     = &aniobj;
        function(spo);
        spo= &junobj;
        function(spo);

        return 0;
}
```

OUTPUT:



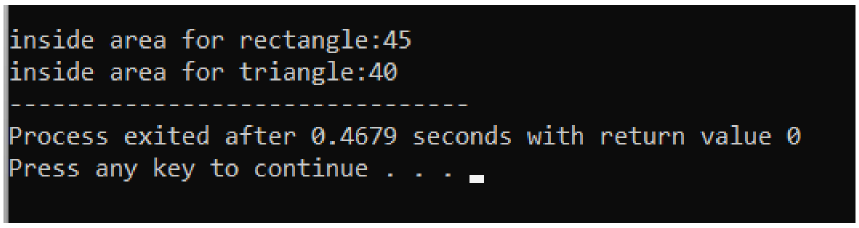
```
hello
print this

--------------------------------
Process exited after 0.1661 seconds with return value 0
Press any key to continue . . .
```

(c) Show the implementation of abstract class

Program:

```
#include<iostream>
using namespace std;
// using abstract methods and classes.
class figure
{
  public:
  double dim1;
  double dim2;
  figure(double a, double b)
{
 dim1 = a;
 dim2 = b;
}
// pure virtual function
virtual double area()=0;
};
class rectangle:public figure
{
  public:
   rectangle(double a, double b):figure(a,b)
{
}
// implement area for rectangle
double area()
{
 cout<<"\ninside area for rectangle:";
  return dim1 * dim2;
}
};
class triangle:public figure
{
 public:
 triangle(double a, double b):figure(a,b)
{
}
// implement area for right triangle

double area()
{
  cout<<"\ninside area for triangle:";
 return dim1 * dim2 / 2;
}
};
int main()
{
 rectangle r(9, 5);
 triangle t(10, 8);
 cout<< r.area();
 cout<< t.area();

 return 0;
}
```

OUTPUT:



```
inside area for rectangle:45
inside area for triangle:40
--------------------------------
Process exited after 0.4679 seconds with return value 0
Press any key to continue . . .
```

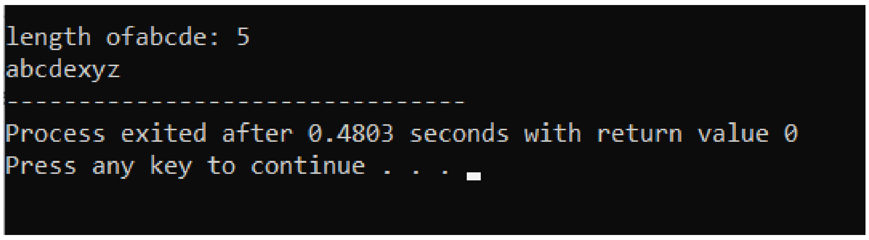
Practical 7.
String handling

(a) String operations for string length, string concatenation

Program:

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
  string str1="abcde";
  string str2="xyz";
  cout<<endl<<"length of"<<str1<<": "<<str1.length();
  string str3=str1+str2;
  cout<<endl<<str3;

}
```

OUTPUT:



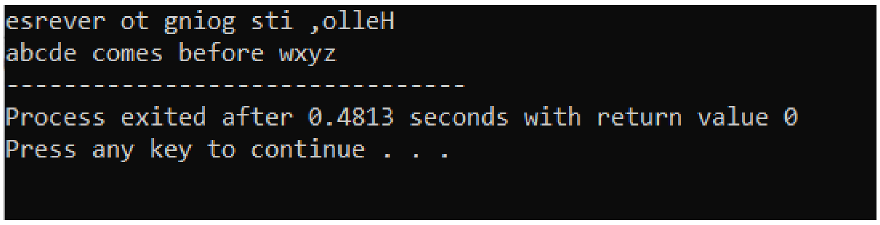
```
length ofabcde: 5
abcdexyz
--------------------------------
Process exited after 0.4803 seconds with return value 0
Press any key to continue . . .
```

(b) string operations for string reverse, string comparison

program:

```
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;
int main()
{
  string str="Hello, its going to reverse";
  reverse(str.begin(), str.end());
  cout<<str;
  string s1="wxyz";
  string s2="abcde";

if(s1<s2)
  cout<<endl<<s1<<"comes before "<<s2;
else
  cout<<endl<<s2<<" comes before "<<s1;

}
```

OUTPUT:



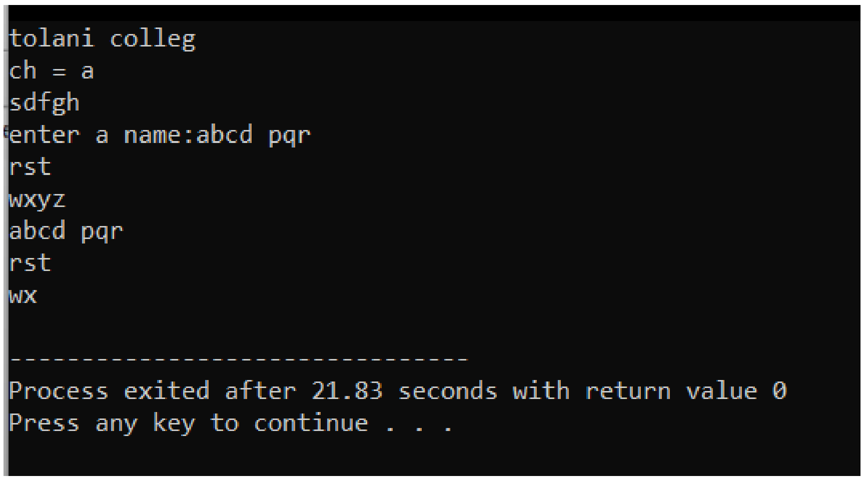
```
esrever ot gniog sti ,olleH
abcde comes before wxyz
--------------------------------
Process exited after 0.4813 seconds with return value 0
Press any key to continue . . .
```

(c) Console formatting function

Program:

```
#include <iostream>
using namespace std;
int main()
{
  char c[] = "tolani college";

  cout.write(c, 13).put('\n'); // put('\n')is used in place of endl
  char ch[] = "a";
  cout<<"ch = ";
  cout.write(ch,1)<<endl; // writes one byte of ch.
  char s[] = "sdfghjk" ;
  cout.write(s, 5)<<endl; // writes 5 bytes from string s
  char name[15];
  cout<< "enter a name:" ;
  cin.read(name ,15); // reads 15 bytes from name
  cout.write(name,15)<<endl; // writes 15 bytes from name

  return 0;
}
```

OUTPUT:



```
tolani colleg
ch = a
sdfgh
enter a name:abcd pqr
rst
wxyz
abcd pqr
rst
wx

--------------------------------
Process exited after 21.83 seconds with return value 0
Press any key to continue . . .
```

Practical 8.
exception handling

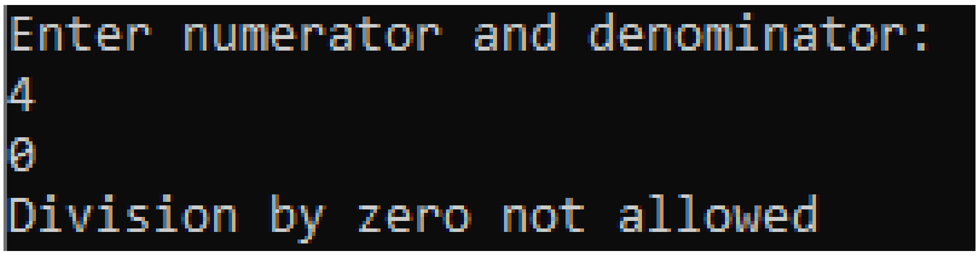
(a) Show the implementation of exception handling.

Program:

OUTPUT:

```
//exception handling program
#include<iostream>
using namespace std;

int main()
{
	int numerator,denominator,result;
	cout<<"Enter numerator and denominator:"<<endl;
	cin>>numerator>>denominator;
	try
	{
		if(denominator==0)
		{
			throw denominator;
		}
		int result=numerator/denominator; //division happens here
	}
	catch(int ex)
	{
		cout<<"Division by zero not allowed"<<endl;
	}


	cout<<"Division is:"<<result;

	return 0;
}
```

OUTPUT:



```
Enter numerator and denominator:
4
0
Division by zero not allowed
```

Practical 9.
File handling

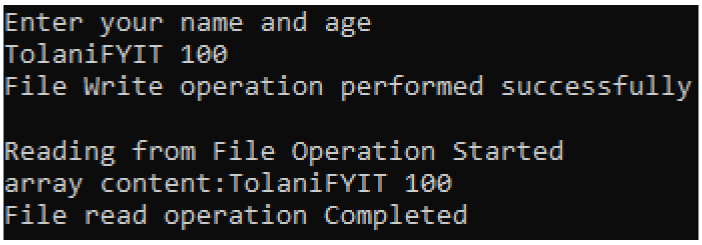
(a) Design a class file demo open a file in read mode and display the total number of words and lines in the file.

Program:

```
//File Handling in C++
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
	//file write operation

	char arr[100]; // character array of size 100 whose name is arr

	cout<<"Enter your name and age"<<endl; // asking the user to enter the details

	//cin>>arr; if we use this..anything after the space is ignored

	cin.getline(arr,100); //to take the entire line as input cin.getline and pass the array
name and size of the array
	// so we are taking the input from the user and will store it in arr

	ofstream myfile("abc.txt"); //to write the file we use ofstream
	//ofstream here will work like datatype like we use int x or float y....so we have user
defined data type which is ofstream and myfile is a name like x and y
	//("abc.txt") is a txt file where will store the content

//   ofstream myfile("abc.txt",ios:: app); // to add the content in the file and keep the previous
content as it is

   //myfile.open(abc.txt); to chk whether the file is open or not
	myfile<<arr; //to write whatever we have in arr in myfile
	myfile.close();
	cout<<"File Write operation performed successfully"<<endl<<endl;

	//file read operation
	cout<<"Reading from File Operation Started"<<endl;
	char arr1[100];
	ifstream obj("abc.txt"); //ifstream to read..this will directly oopen the file in read
mode.
	obj.getline(arr1,100); //obj is lined with file....whatevr the file content it is going to br
filled inside the array variable
	cout<<"array content:"<<arr1<<endl;
	cout<<"File read operation Completed"<<endl;
	obj.close();
	return 0;
```

}

OUTPUT:



```
Enter your name and age
TolaniFYIT 100
File Write operation performed successfully

Reading from File Operation Started
array content:TolaniFYIT 100
File read operation Completed
```

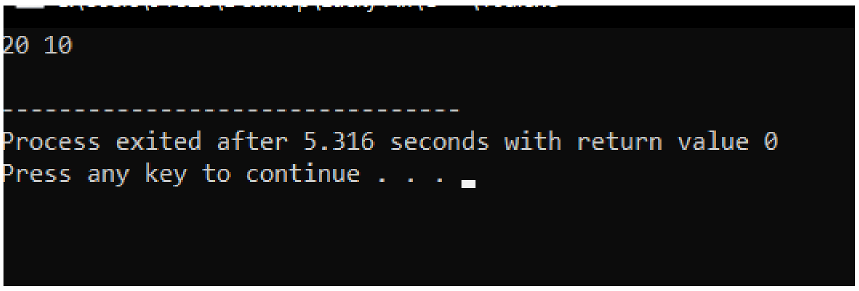
Practical 10.
Templates

(a) Show the implementation of template class library for swap function.

Program:

```
#include <iostream>
using namespace std;
template <class T>
int swap_numbers(T& x, T& y)
{
        T t;
        t = x;
        x = y;
        y = t;
        return 0;
}
int main()
{
        int a, b;
        a = 10, b = 20;
        swap_numbers(a, b);
        cout << a << " " << b << endl;
        return 0;
}
```

OUTPUT:



```
20 10

--------------------------------
Process exited after 5.316 seconds with return value 0
Press any key to continue . . .
```

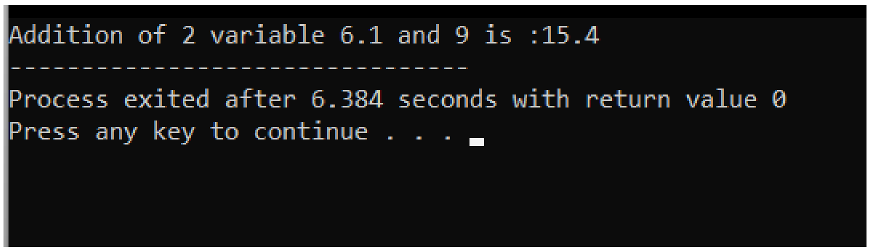
(b) Write a Program using Function Template.

Program:

```
#include<iostream>
using namespace std;
template<typename T, typename U>
U add ( T x, U y)
{
   return (x+y);
}
int main(){

   cout<<"Addition of 2 variable 6.1 and 9 is :"<<add<int>(6,9.4);
```

```
        return 0;
}
```

OUTPUT:

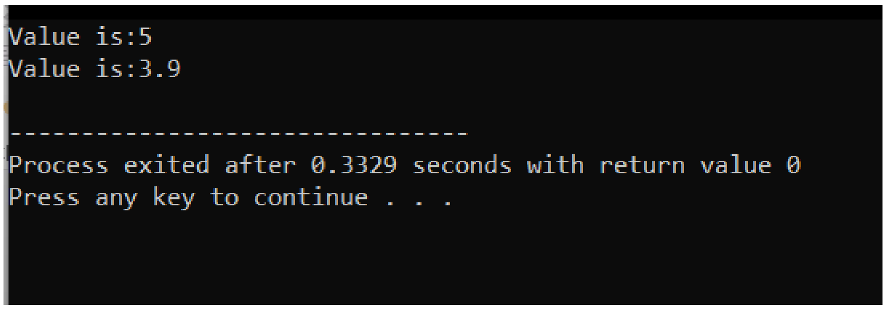


```
Addition of 2 variable 6.1 and 9 is :15.4
--------------------------------
Process exited after 6.384 seconds with return value 0
Press any key to continue . . .
```

(c) Write a Program Using Class Template.
Program:

```
#include<iostream>
using namespace std;
template<typename T>
class weight
   {
   private:
      T kg;
   public:
   void setData(T x)
   {
      kg=x;
   }
   T getData()
     {
      return kg;
   }
   };
   int main()
   {
     weight<int>obj1;
     obj1.setData(5);
     cout<<"Value is:"<<obj1.getData()<<endl;

       weight<float>obj2;
     obj2.setData(3.9);
     cout<<"Value is:"<<obj2.getData()<<endl;

     return 0;
   }
```

OUTPUT:



```
Value is:5
Value is:3.9

--------------------------------
Process exited after 0.3329 seconds with return value 0
Press any key to continue . . .
```