Term work

of

OOPs Using C++ Lab (PCS-307)

Submitted in partial fulfillment of the requirement for the III semester

Bachelor of Technology

By

Shreya Paithani

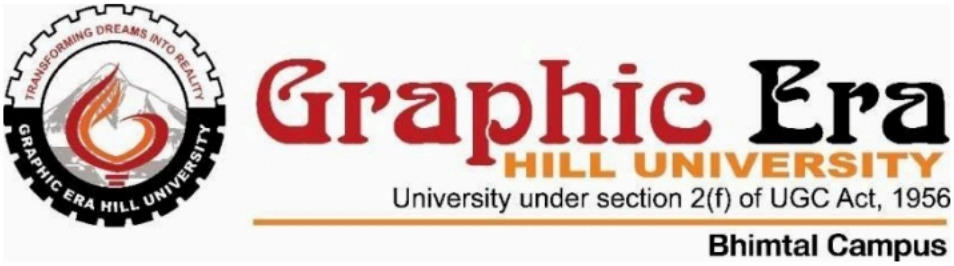
2261647

Under the Guidance of

Ms. Kashish Mirza

Lecturer

Dept. of CSE



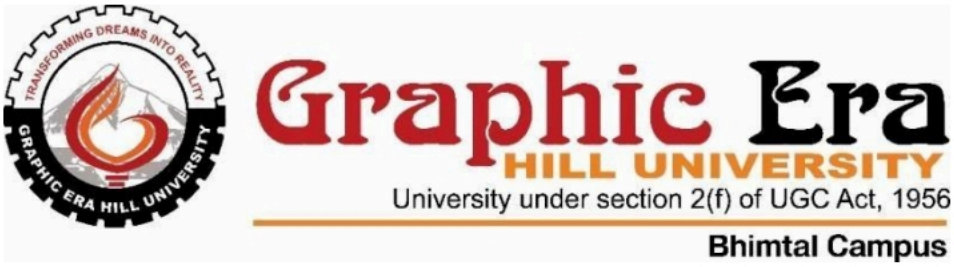
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI

DISTRICT- NAINITAL-263132

2023-2024

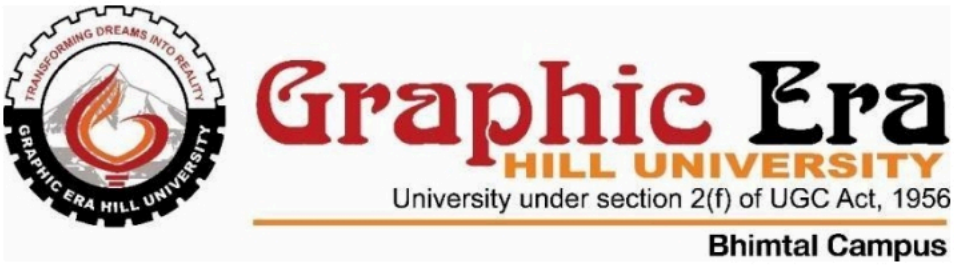


CERTIFICATE

The term work of OOPs with C++ Lab (PCS-307), being submitted by Shreya Paithani d/o Mr. Suraj Singh, University Roll Number 2261647 to Graphic Era Hill University Bhimtal Campus for the award of bona fide work carried out by her. She has worked under my guidance and supervision and fulfilled the requirement for the submission of this work report.

(Ms. Kashish Mirza) **(Mr. Ankur Singh Bisht)**

Lecturer **HOD, CSE Dept.**



ACKNOWLEDGEMENT

I take immense pleasure in thanking Honorable **Ms. Kashish Mirza** (Lecturer, CS, GEHU Bhimtal Campus) for allowing us to carry out this project work under his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance and useful suggestions that have helped me in developing my subject concepts as a student.

I want to extend thanks to our President **"Prof. (Dr.) Kamal Ghanshala"** for providing us all infrastructure and facilities to work in need without which this work would not be possible.

(SHREYA PAITHANI)

Stl.1463shreya11@gmail.com

Graphic Era
HILL UNIVERSITY
University under section 2(f) of UGC Act, 1956
Bhimtal Campus

STUDENT'S DECLARATION

I,**Shreya Paithani**, hereby declare the work, which is being presented in the report, entitled **Term work of OOPs with C++ Lab (PCS-307)** in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (Computer Science)** in the session **2023-2024** for semester III, is an authentic record of my own work carried out under the supervision of **Ms. Kashish Mirza** (Graphic Era Hill University, Bhimtal)

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date: ……

……

(Full signature of student)

Graphic Era HILL UNIVERSITY
University under section 2(f) of UGC Act, 1956
Bhimtal Campus

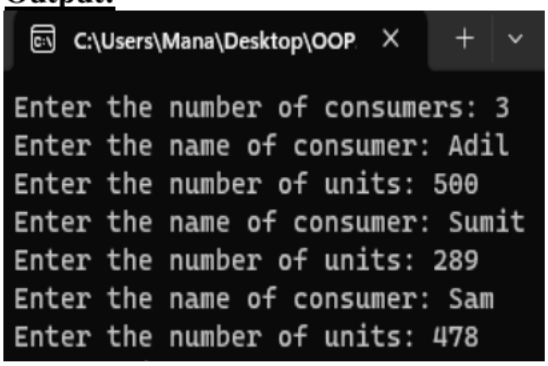
Computer Science and Engineering Department
OOPS LAB (PCS-307)

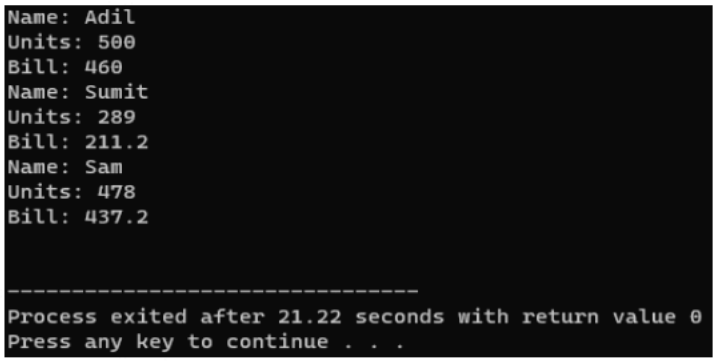
S No.	Date	List Of Experiments

1. An electricity board charges the following rates to domestic users to discourage large consumption of energy. a. For the first 100 units: - 60 P per unit b. For the next 200 units: - 80 P per unit c. Beyond 300 units: - 90 P per unit . All users are charged a minimum of Rs 50 if the total amount is more than Rs 300then an additional surcharge of 15% is added. Implement a C++ program to readthe names of users and number of units consumed and display the charges with names.

```
#include<iostream>
 using namespace std;
 class bill
{ private:
  string name; int units; double total;
  public:
  void input(); void display(); void calculate();
};
void bill::input()
{ cout<<"Enter the name of consumer: "; cin>>name;
  cout<<"Enter the number of units: "; cin>>units;}
void bill::display()
{ cout<<"Name: "<<name<<"\n"; cout<<"Units: "<<units<<"\n";
   cout<<"Bill: "<<total<<"\n";}
void bill::calculate()
{ if(units<=100)
    { total=units*0.60;}
  else if(units>100&&units<=300)
    { total=(0.6*100)+((units-100)*0.8);}
  else if(units>300)
    { total=(100*0.6)+(200*0.8)+((units-300)*0.90);}
  else
    {total=50;}
   if(total<50)
    { total=50;}
  if(total>300)
    { int charge=(total*0.15); total=total+charge;}
}
int main()
{ int n,i;
  cout<<"Enter the number of consumers: "; cin>>n;
  bill arr[n]; for(i=0;i<=n-1;i++)
  { arr[i].input();
    arr[i].calculate();}
  for(i=0;i<=n-1;i++) arr[i].display(); cout<<endl;
  return 0; }
```

Output:



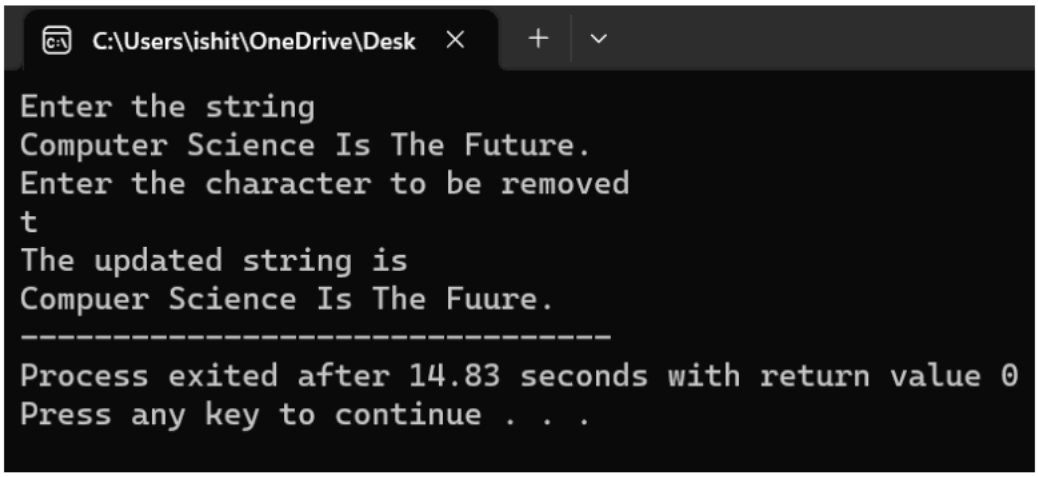


2. Construct a C++ program that removes a specific character from a given string and return the updated string.

Typical Input: computer science is the future
Typical Output: compuer science is he fuure

```
#include<iostream>
using namespace std;
int main()
{
   strings1,s2="";
   char a;
   int i;
   cout<<"Enter the string"<<endl;
   getline(cin,s1);
   cout<<"Enter the character to be removed"<<endl;
   cin>>a;
   for(i=0;i<=s1.length();i++)
   {
      if(s1[i]==a)
      {
         continue;
      }
      else
      {
         s2=s2+s1[i];

      }
   }
   cout<<"The updated string is"<<endl<<s2;
   return 0;
}
```

Output:

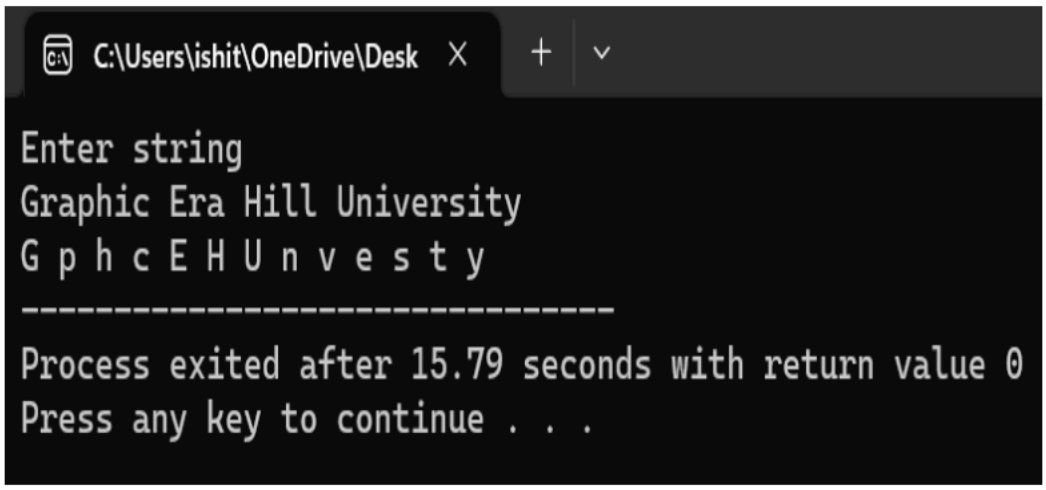


3. Implement a C++ program to find the non-repeating characters in string.Typical Input: graphic era university

Typical Output: c g h n p s t u v y

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string s;
    cout<<"Enter string \n";
    getline(cin,s);
    for(int i=0;i<s.size();i++)
    {
        int duplicate=0;
        for(int j=0;j<s.size();j++)
        {
            if( s[i]==s[j] and i!=j)
            {
                duplicate=1;
                break;
            }
        }
        if(duplicate==0)
        {
            cout<<s[i]<<" ";
        }
    }
}
```

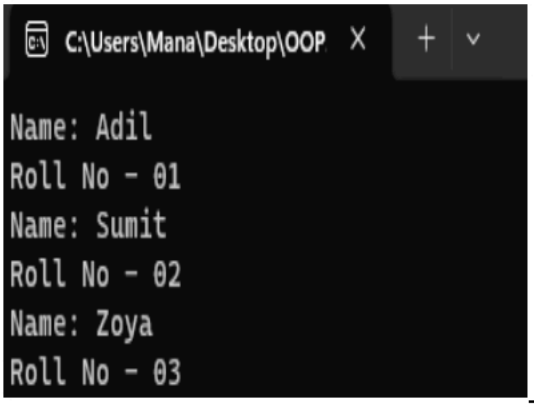
Output:



4.Implement a C++ program to demonstrate the concept of data abstraction using the concept of Class and Objects.

```
#include<iostream>
using namespace std;
class student
{
   string name;
   int rn;
   static char sec;
   public:
      void input();
      static void changesec(char newsec);
      void display();
};
char student::sec;
void student::input()
{cout<<"Name: "; cin>>name; cout<<"Roll No - "; cin>>rn;}
void student::changesec(char newsec)
{sec=newsec;}
void student::display()
{cout<<"Name:"<<name<<endl<<"Roll No-
"<<rn<<endl<<"Section:"<<sec<<endl;}
int main()
{
   student obj1,obj2,obj3;
   obj1.input();
   obj2.input();
   obj3.input();
   student::changesec('C');
   obj1.display();
   obj2.display();
   obj3.display();
   return 0;
}
```

Output:



5. Define a class Hotel in C++ with the following specifications.
Private members:

- **Rno Data member to store room number**
- **Name Data member to store customer name**
- **Tariff Data member to store per day charges**
- **NOD Data member to store number of days of stay**
- **CALC() Function to calculate and return amount as NOD*Tariff, and if the value of days*Tariff >10000, then total amount is 1.05*days*Tariff.**

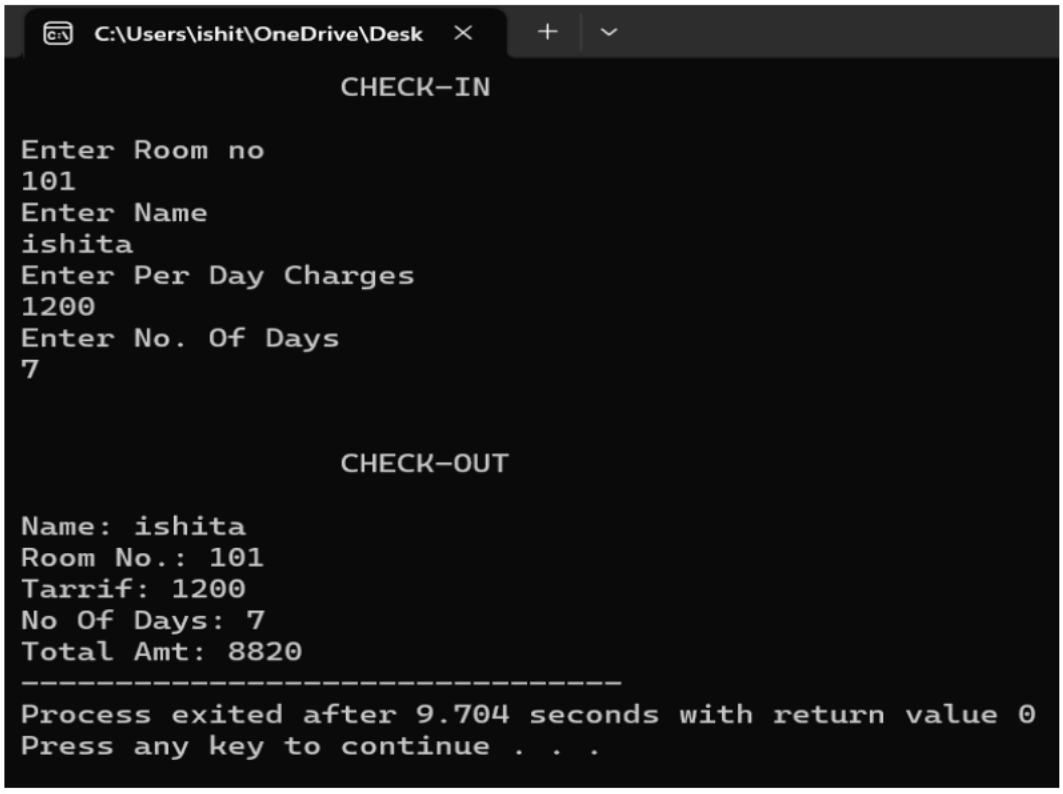
Public members:

- **Checkin() Function to enter the content Rno, Name, Tariff and NOD**
- **Checkout() Function to display Rno, Name, Tariff, NOD and Amount (amount to be displayed by calling function) CALC()**

```
#include<iostream>
using namespace std;
class Hotel
{
   int rno,nod;
   string name;
   float tarrif;
   public:
      void Checkin();
        void Checkout();
      int CALC();
 };
     int main()
     {
   Hotel obj1;
   cout<<"                  CHECK-IN          ";
   obj1.Checkin();
   cout<<"\n\n                    CHECK-OUT         ";
   obj1.Checkout();
   obj1.CALC();
    }

     void Hotel:: Checkin()
     {
   cout<<"\n\nEnter Room no"<<endl;
   cin>>rno;
   cout<<"Enter Name"<<endl;
   cin>>name;
   cout<<"Enter Per Day Charges"<<endl;
   cin>>tarrif;
   cout<<"Enter No. Of Days\n";
```

```
cin>>nod;
  }

  void Hotel ::Checkout()
 {
  cout<<"\n\nName: "<<name<<endl;
  cout<<"Room No.: "<<rno<<endl;
  cout<<"Tarrif: "<<tarrif<<endl;
  cout<<"No Of Days: "<<nod<<endl;
 }

 int Hotel::CALC()
  {
     int amt=0;
     if(nod*tarrif>1000)
     amt=nod*tarrif*1.05;
     else
     amt=nod*tarrif;
     cout<<"Total Amt: "<<amt;

  }
```

Output:



6. **Implement a Program in C++ by defining a class to represent a bank account. Include the following:**

Data Members:

- **Name of the depositor**
- **Account number**
- **Type of account (Saving, Current etc.)**
- **Balance amount in the account**

Member Functions:

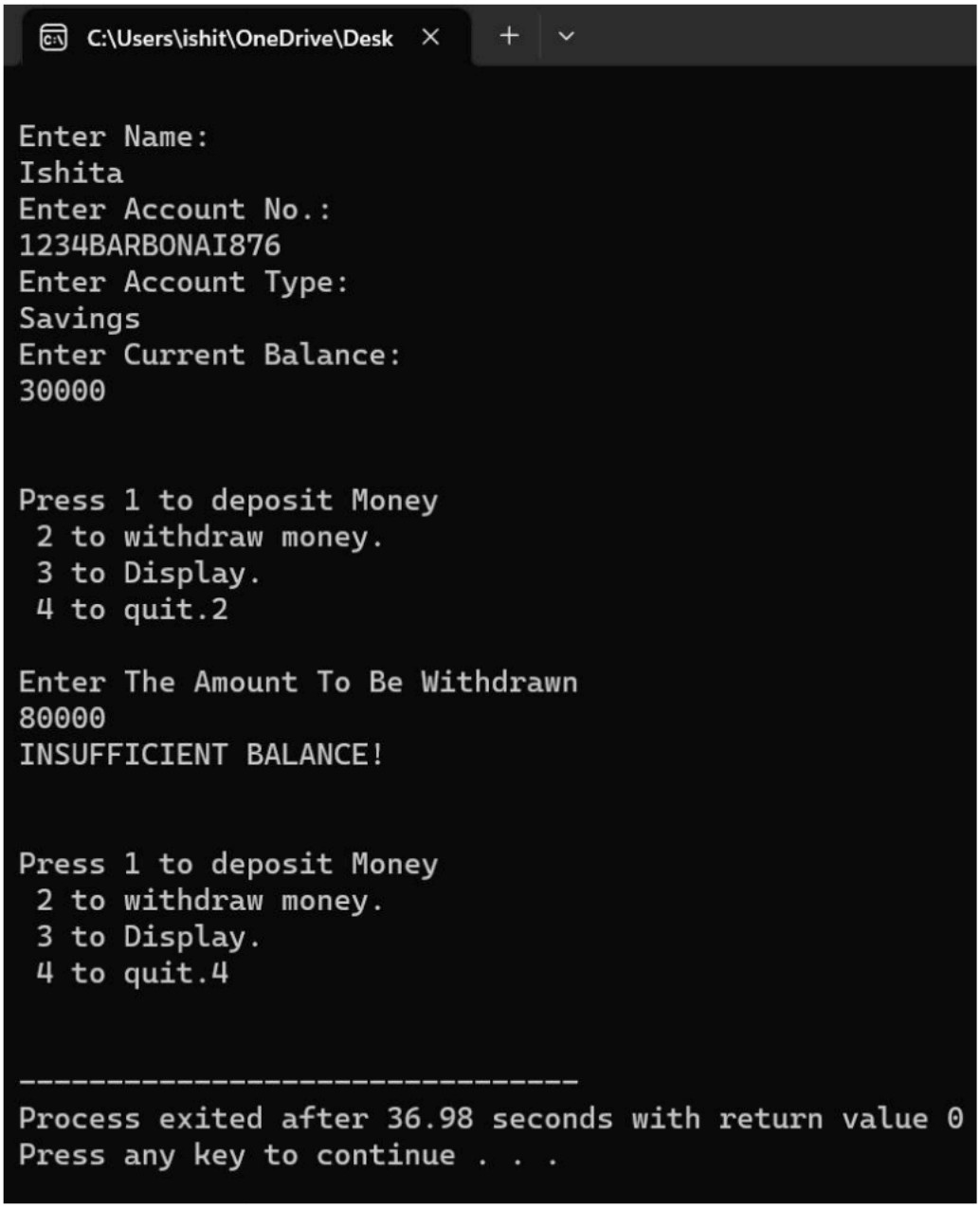
- **To assign initial values**
- **To deposit an amount**
- **To withdraw an amount after checking the balance**
- **To display name and balance**

```
#include<iostream>
using namespace std;
class bank
{
   string name;
   long acn;
   string type;
   float bal;
   public:
      void input();
      void deposit();
      void withdraw();
      void disp();
};
void bank::input()
{
   cout<<"Name: ";
   cin>>name;
   cout<<"Account Number: ";
   cin>>acn;
   cout<<"Type of account: ";
   cin>>type;
   cout<<"Enter the balance amount: ";
   cin>>bal;
}
void bank::deposit()
{
   float amt;
   cout<<"Enter the amount to be deposited: ";
   cin>>amt;
   bal=bal+amt;
```

```
   cout<<"Amount deposited successfully."<<endl;
}
void bank::withdraw()
{
   float amt;
   cout<<"Enter the amount to be withdrawn: ";
   cin>>amt;
   if(amt>bal)
   {
      cout<<"Insufficient Balance"<<endl;
      return;
   }
   else
   {
      bal=bal-amt;
      cout<<"Amount withdrawn successfully."<<endl;
   }
}
void bank::disp()
{
   cout<<"Name: "<<name<<endl<<"Balance: "<<bal<<endl;
}
int main()
{
   int choice;
   bank obj1;
   obj1.input();
   while(choice!=4)
   {
      cout<<"Press 1 to deposit"<<endl<<"Press 2 to
withdraw"<<endl<<"Press 3 to display balance"<<endl<<"Press 4
to quit"<<endl;
      cin>>choice;
      switch(choice)
      {
         case 1:
            obj1.deposit();
            break;
         case 2:
            obj1.withdraw();
            break;
         case 3:
            obj1.disp();
            break;
         case 4:
            return 0;
```

```
            default:
               cout<<"Invalid choice"<<endl;
         }
      }
}
```

Output:



7. Anna is a contender for valedictorian of her high school. She wants to know how many students (if any) have scored higher than her in the exams given during this semester.

Create a class named Student with the following specifications:

- **An instance variable named scores holds a student's 5 exam scores.**
- **A void input () function reads 5 integers and saves them to scores.**
- **An int calculateTotalScore() function that returns the sum of the student's scores.**

Input Format

In the void Student::input() function, you must read 5 scores from standard input and save them to your scores instance variable.

Output Format

In the int Student::calculateTotalScore() function, you must return the student's total grade (the sum of the values in scores).

The code in the editor will determine how many scores are larger than Anna's and print that number to the console.

Sample Input

The first line contains n, the number of students in Anna's class. The n subsequent lines contain each student's 5 exam grades for this semester.

3
30 40 45 10 10
40 40 40 10 10
50 20 30 10 10

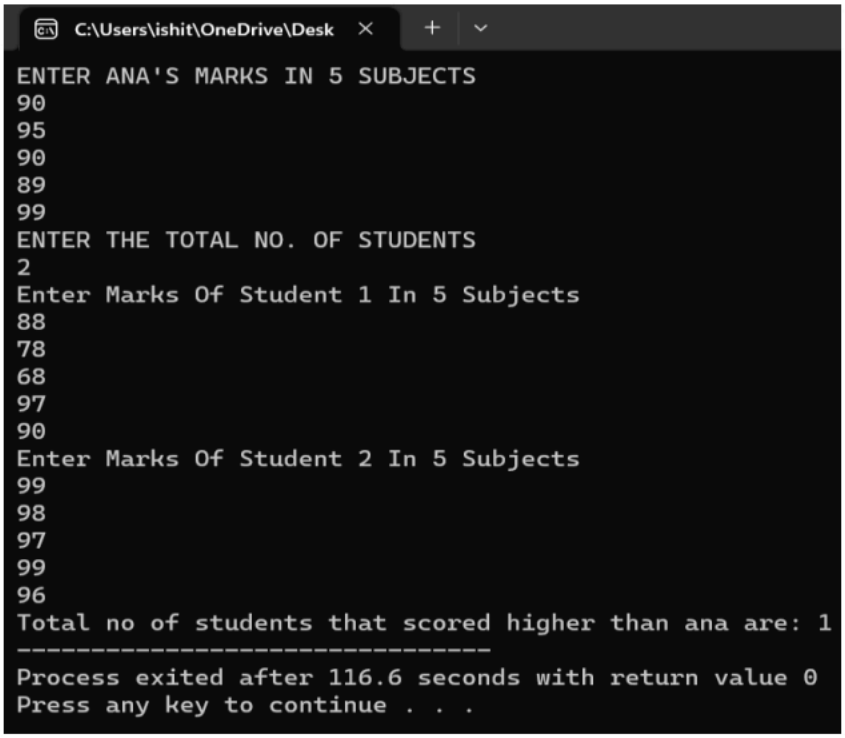
Sample Output

1

```
#include<iostream>
using namespace std;
class student
{
   public:
      double score[5];
   public:
   void input();
   int calculateTotalScore();
};
 int main()
 {  int t;
     student ana;
   cout<<"ENTER ANA'S MARKS IN 5 SUBJECTS\n";
   ana.input();
   t=ana.calculateTotalScore();
   int n,tot,c=0;
   cout<<"ENTER THE TOTAL NO. OF STUDENTS\n";
   cin>>n;
```

```
    student obj[n];
    for(int i=1;i<=n;i++)
    {
      cout<<"Enter Marks Of Student "<<i<<" In 5 Subjects\n";
      obj[i].input();
      tot=obj[i].calculateTotalScore();
      if(t<tot)
      c++;
    }
    cout<<"Total no of students that scored higher than ana
are: "<< c;
 }
 void  student::input()
 {
   for(int i=0;i<5;i++)
   {
      cin>>score[i];
    }
 }
  int student::calculateTotalScore()
 { double sum=0;
   for(int i=0;i<5;i++)
    sum+=score[i];
    return sum;
 }
```

Output:



8. **Construct a Program in C++ to show the working of function overloading (compile time polymorphism) by using a function named calculate Area () to calculate area of square, rectangle and triangle using different signatures as required.**

```
#include<iostream>
#include<cmath>
using namespace std;
class poly
{
   public:
      float l,b,s,x,y,z;
      void area(float,float);
      void area(float);
      void area(float,float,float);
};
int main()
{
   poly p;
   int c;
     while(c!=4)
     {
      cout<<"\n\nENTER-\n1 For Square\n2 For Rectangle\n3 For
Triangle.\n4 To Quit\n";
     cin>>c;
     switch (c)
     {
      case 1:
        cout<<"Enter The Size Of Sides\n";
     cin>>p.s;
     p.area(p.s);
     break;
     case 2:
     cout<<"Enter The Length And The Breath\n";
     cin>>p.l>>p.b;
     p.area(p.l,p.b);
     break;
    case 3:
    cout<<"Enter The Three Sides Of The Triangle\n";
      cin>>p.x>>p.y>>p.z;
     p.area(p.x,p.y,p.z);
     break;
    case 4:
    cout<<endl;
    break;
```

```
    default:
    cout<<"INVALID CHOICE!\n";
    break;
    }
   }
}
void poly::area(float s)
   cout<<"The Area Of Square is: "<<s*s<<endl;
void poly::area(float l,float b)
{
   cout<<"The Area Of The Rectangle is: "<<l*b;
}
void poly::area(float x,float y,float z)
{
   float s,ar,a,b,c;
   s=(x+y+z)/2.0;
   ar=sqrt(s*(s-x)*(s-y)*(s-z));
    cout<<"Area Of The Triangle is: "<<ar;
 }
```

Output:

```
C:\Users\ishit\OneDrive\Desk
ENTER-
1 For Square
2 For Rectangle
3 For Triangle.
4 To Quit
1
Enter The Size Of Sides
7.89
The Area Of Square is: 62.2521


ENTER-
1 For Square
2 For Rectangle
3 For Triangle.
4 To Quit
2
Enter The Length And The Breath
16.9
12.908
The Area Of The Rectangle is: 218.145

ENTER-
1 For Square
2 For Rectangle
3 For Triangle.
4 To Quit
4


--------------------------------
Process exited after 27.3 seconds with return value 0
```

9. **Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance.**

Data Members -

- **partNumber (type String)**
- **partDescription (type String)**
- **quantity of the item being purchased (type int)**
- **price_per_item (type double)**

Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named getInvoiceAmount() that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named invoiceTest that demonstrates class Invoice's capabilities.

```
#include<iostream>
using namespace std;
class invoice
{
   string part_no;
   string part_des;
   int qty;
   double ppi;
   public:
      invoice();
      void set_part_no();
      void set_part_des();
      void set_qty();
      void set_ppi();
      string get_part_no();
      string get_part_des();
      int get_qty();
      double get_ppi();
      double get_invoice_amount();
};
invoice::invoice()
{
   cout<<"Enter part number: ";
   cin>>part_no;
   cout<<"Enter description of part: ";
   cin>>part_des;
   cout<<"Enter quantity of the item being purchased: ";
   cin>>qty;
   cout<<"Enter price per item: ";
   cin>>ppi;
```

```
}
void invoice::set_part_no()
{
   cout<<"Enter part number: ";
   cin>>part_no;
}
void invoice::set_part_des()
{
   cout<<"Enter description of part: ";
   cin>>part_des;
}
void invoice::set_qty()
{
   cout<<"Enter quantity of the item being purchased: ";
   cin>>qty;
}
void invoice::set_ppi()
{
   cout<<"Enter price per item: ";
   cin>>ppi;
}
string invoice::get_part_no()
{
   return part_no;
}
string invoice::get_part_des()
{
   return part_des;
}
int invoice::get_qty()
{
   return qty;
}
double invoice::get_ppi()
{
   return ppi;
}
double invoice::get_invoice_amount()
{
   if(qty<0)
   {
      qty=0;
   }
   if(ppi<0)
   {
      ppi=0;
```

```
    }
    return (qty*ppi);
}
int main()
{
    invoice test;
    //set methods
    cout<<"Set Methods:\n";
    test.set_part_no();
    test.set_part_des();
    test.set_qty();
    test.set_ppi();
    //get methods
    cout<<"Get Methods:\n";
    string a,b; int c; double d;
    a=test.get_part_no();
    cout<<a;
    b=test.get_part_des();
    cout<<b;
    c=test.get_qty();
    cout<<c;
    d=test.get_invoice_amount();
    cout<<d;
    return 0;
}
```

Output:

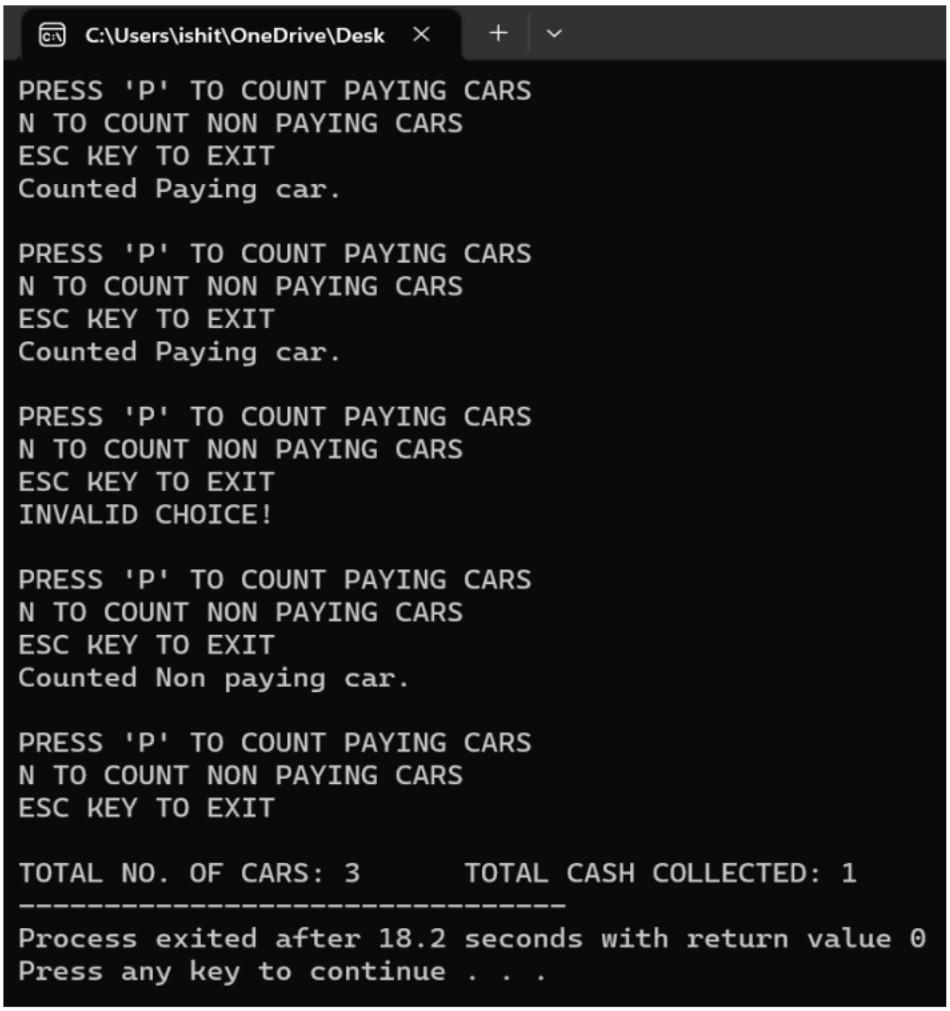
```
Enter part number: 1205
Enter description of part: Ghj
Enter quantity of the item being purchased: 92
Enter price per item: 31
Set Methods:
Enter part number: 1520
Enter description of part: Wer
Enter quantity of the item being purchased: 96
Enter price per item: 56.3
Get Methods:
1520
Wer
96
5404.8

--------------------------------
Process exited after 33.47 seconds with return value 0
Press any key to continue . . .
```

10. **Imagine a tollbooth with a class called TollBooth. The two data items are of type unsigned int and double to hold the total number of cars and total amount of money collected. A constructor initializes both of these data members to 0. A member function called payingCar( )increments the car total and adds 0.5 to the cash total. Another function called nonPayCar( ) increments the car total but adds nothing to the cash total. Finally a member function called display( )shows the two totals. Include a program to test this class. This program should allow the user to push one key to count a paying car and another to count a non paying car. Pushing the ESC key should cause the program to print out the total number of cars and total cash and then exit.**

```
#include<iostream>
#include<conio.h>
using namespace std;
class tollbooth
{
   unsigned int n;
    double m;
   public:
   tollbooth();
   void payingCar();
   void nonPayCar();
   void display();
};
int main()
{
   tollbooth booth;
   char ch;
   do {
    cout<<"PRESS 'P' TO COUNT PAYING CARS\nN TO COUNT NON
PAYING CARS\nESC KEY TO EXIT\n";
    ch=_getch();
      switch(ch)
      {case 'p':
      case 'P':
          booth.payingCar();
          cout<<"Counted Paying car.\n\n";
          break;
      case'N':
      case'n':
           booth.nonPayCar();
           cout<<"Counted Non paying car.\n\n";
           break;
      default:
           if(ch!=27)
```

```
            cout<<"INVALID CHOICE! \n"<<endl;
            break;
      }
        if(ch==(char)27)
        booth.display();
   }while(ch!=27);
}
tollbooth::tollbooth()
{ n=0;m=0.0;}
void tollbooth::payingCar()
{n++; m=m+0.5;}
void tollbooth::nonPayCar()
{ n++; }
void tollbooth::display()
{
 cout<<"\nTOTAL NO. OF CARS: "<<n ;
 cout<<"      TOTAL CASH COLLECTED: "<<m;
}
```

Output:

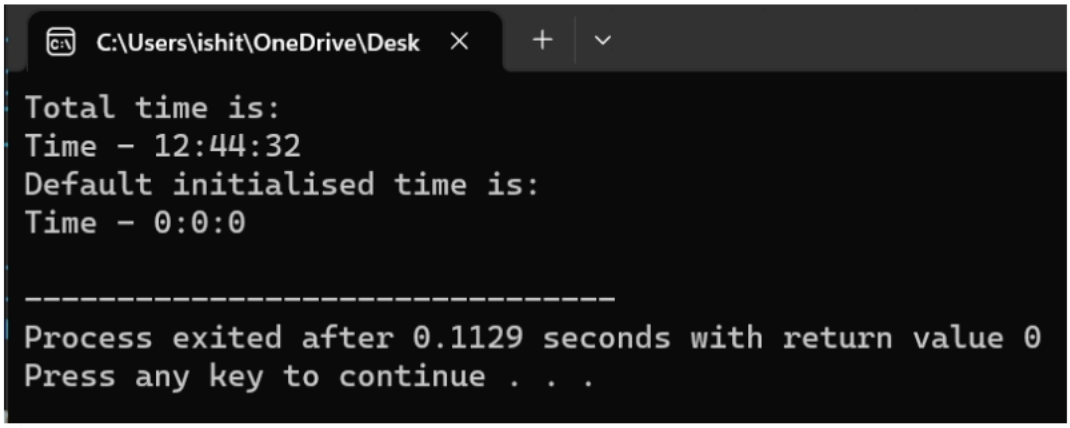


11. Create a class called Time that has separate int member data for hours, minutesand seconds. One constructor should initialize this data to 0, and another shouldinitialize it to fixed values. A member function should display it in 11:59:59 format. A member function named add() should add two objects of type time passed as arguments. A main( ) program should create two initialized values together, leaving the result in the third time variable. Finally, it should display the value of this third variable.

```
#include<iostream>
using namespace std;
class time
{
   int hours;
   int min;
   int sec;
   public:
      time();
      time(int,int,int);
      void display();
      time add(time);
};
time::time()
{
   hours=0;
   min=0;
   sec=0;
}
time::time(int a,int b,int c)
{
   hours=a;
   min=b;
   sec=c;
   if(sec>60)
   {
      min=min+(sec/60);
      sec=sec%60;
   }
   if(min>60)
   {
      hours=hours+(min/60);
      min=min%60;
   }
}
void time::display()
{
   cout<<"Time - "<<hours<<":"<<min<<":"<<sec<<endl;
```

```
}
time time::add(time a)
{
   time t3;
   t3.hours=hours+a.hours;
   t3.min=min+a.min;
   t3.sec=sec+a.sec;
   if(t3.sec>60)
   {
      t3.min=t3.min+(t3.sec/60);
      t3.sec=t3.sec%60;
   }
   if(t3.min>60)
   {
      t3.hours=t3.hours+(t3.min/60);
      t3.min=t3.min%60;
   }
   return t3;
}
int main()
{
   int a1,a2,b1,b2,c1,c2;
   time t1(2,32,27),t2(10,12,05),t3,t4;
   t3=t1.add(t2);
   cout<<"Total time is: "<<endl;
   t3.display();
   cout<<"Default initialised time is: "<<endl;
   t4.display();
   return 0;
}
```

Output:



12. Create class SavingsAccount. Use a static variable annualInterestRate to store the annual interest rate for all account holders. Each object of the class contains a private instance variable savingsBalance indicating the amount the saver currently has on deposit. Provide method calculateMonthlyInterest() to calculatethe monthly interest by multiplying the savingsBalance by annualInterestRate divided by 12.This interest should be added tosavingsBalance. Provide a static method modifyInterestRate() that sets the annualInterestRate to a new value. Write a program to test class SavingsAccount. Instantiate two savingsAccount objects, saver1 and saver2, with balances of Rs2000.00 and Rs3000.00, respectively. Set annualInterestRate to 4%, then calculate the monthly interest and print the new balances for both savers. Then set the annualInterestRate to 5%, calculate the next month's interest and print the new balances for both savers.

```
#include<iostream>
using namespace std;
class SavingsAccount
{
   static double air;
   double s_bal;
   public:
      void i_bal();
      void calculateMonthlyIntrest();
      static void modifyIntrestRate();
      void disp();
};
double SavingsAccount::air;
void SavingsAccount::i_bal()
{
   cout<<"Enter balance: ";
   cin>>s_bal;
}
void SavingsAccount::calculateMonthlyIntrest()
{
   double intrest=(s_bal*air)/12.0;
   double u_bal=s_bal+intrest;
   s_bal=u_bal;
}
void SavingsAccount::modifyIntrestRate()
{
   double intrest;
   cout<<"Enter new value of annual intrest rate: ";
   cin>>intrest;
   air=intrest;
}
void SavingsAccount::disp()
{
```

```
   cout<<"Updated Balance: "<<s_bal<<endl;
}
int main()
{
   int n,i,ch;
   cout<<"Enter the number of accounts: ";
   cin>>n;
   SavingsAccount saver[n];
   while(ch!=4)
   {
      cout<<"Press: \n1 to enter initial balance.\n2 to set
annual intrest rate.\n3 to display new balance\n4 to quit\n";
      cin>>ch; cout<<endl;
      switch(ch)
      {
         case 1:
            for(i=0;i<n;i++)
            saver[i].i_bal();
            break;
         case 2:
            SavingsAccount::modifyIntrestRate();
            for(i=0;i<n;i++)
            saver[i].calculateMonthlyIntrest();
            break;
         case 3:
            for(i=0;i<n;i++)
            saver[i].disp();
            break;
         case 4:
            return 0;
         default:
            cout<<"Invalid choice\n\n";
            break;
      }
   }
}
```

Output:

```
Enter the number of accounts: 2
Press:
1 to enter initial balance.
2 to set annual intrest rate.
3 to display new balance
4 to quit
1
Enter balance: 2000
Enter balance: 3000
Press:
1 to enter initial balance.
2 to set annual intrest rate.
3 to display new balance
4 to quit
2
Enter new value of annual intrest rate: 4
Press:
1 to enter initial balance.
2 to set annual intrest rate.
3 to display new balance
4 to quit
3
Updated Balance: 2666.67
Updated Balance: 4000
Press:
1 to enter initial balance.
2 to set annual intrest rate.
3 to display new balance
4 to quit
2
Enter new value of annual intrest rate: 5
Press:
1 to enter initial balance.
2 to set annual intrest rate.
3 to display new balance
4 to quit
3
Updated Balance: 3777.78
Updated Balance: 5666.67
Press:
1 to enter initial balance.
2 to set annual intrest rate.
3 to display new balance
4 to quit
4

--------------------------------
Process exited after 22.69 seconds with return value 0
Press any key to continue . . .
```

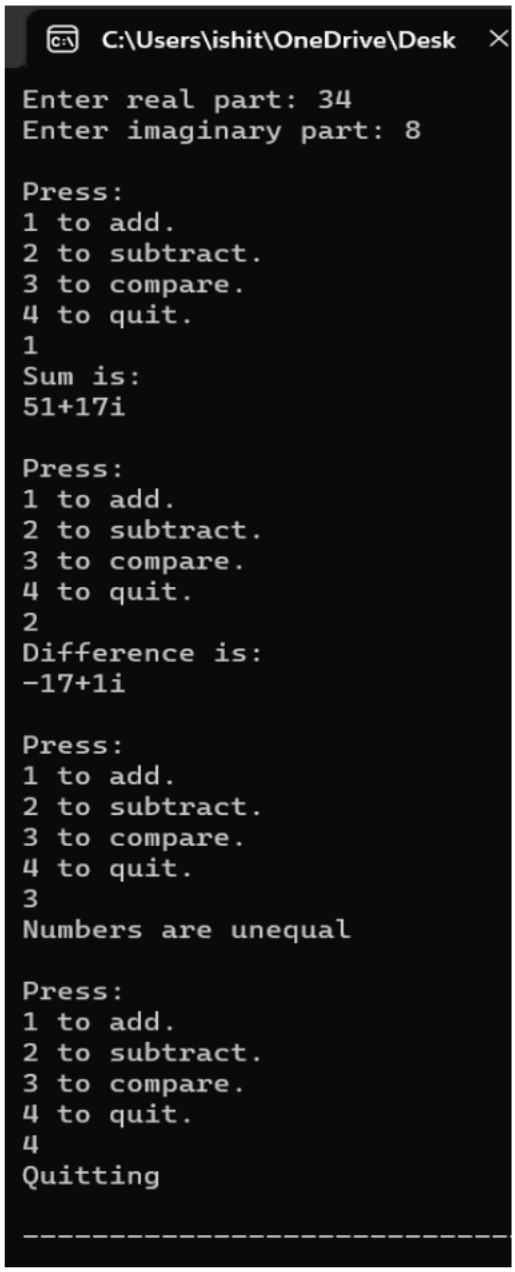
13.Create a class Complex having two int type variable named real & img denoting real and imaginary part respectively of a complex number. Overload +, -, == operator to add, to subtract and to compare two complex numbers being denotedby the two complex type objects.

```
#include<iostream>
using namespace std;
class complex
{
   private:
      int real;
      int img;
   public:
      void input();
      void display();
      complex operator+(complex);
      complex operator-(complex);
      int operator==(complex);
};
void complex::input()
{
   cout<<"Enter real part: ";
   cin>>real;
   cout<<"Enter imaginary part: ";
   cin>>img;
   cout<<"\n";
}
void complex::display()
{
   if(img<0)
   { cout<<real<<"-"<<-img<<"i"; }
   else
   { cout<<real<<"+"<<img<<"i";}
   cout<<"\n\n";
}
complex complex::operator+(complex obj2)
{
   complex obj3;
   obj3.real=real+obj2.real;
   obj3.img=img+obj2.img;
   return obj3;
}
```

```
complex complex::operator-(complex obj2)
 {
    complex obj3;
    obj3.real=real-obj2.real;
    obj3.img=img-obj2.img;
    return obj3;
 }
 int complex::operator==(complex obj1)
 {
    if(real==obj1.real||img==obj1.img)
    {
       return 1;
    }
    return 0;
 }
 int main()
 {
    complex a1,a2,a3;
    int c,d;
    a1.input();
    a2.input();
    while(c!=4)
    {
       cout<<"Press: \n1 to add.\n2 to subtract.\n3 to
 compare.\n4 to quit.\n";
       cin>>c;
       switch(c)
       {
          case 1:
             a3=a1+a2;
             cout<<"Sum is:\n";
             a3.display();
             break;
          case 2:
             a3=a1-a2;
             cout<<"Difference is:\n";
             a3.display();
             break;
          case 3:
             d=a1==a2;
             if(d==1)
             {
                cout<<"Numbers are equal\n\n";
             }
             else
             {
```

```
                cout<<"Numbers are unequal\n\n";
            }
            break;
        case 4:
            cout<<"Quitting\n";
            return 0;
        default:
            cout<<"Invalid choice\n\n";
    }
  }
}
```

Output:



14.Using the concept of operator overloading. Implement a program to overload the following:

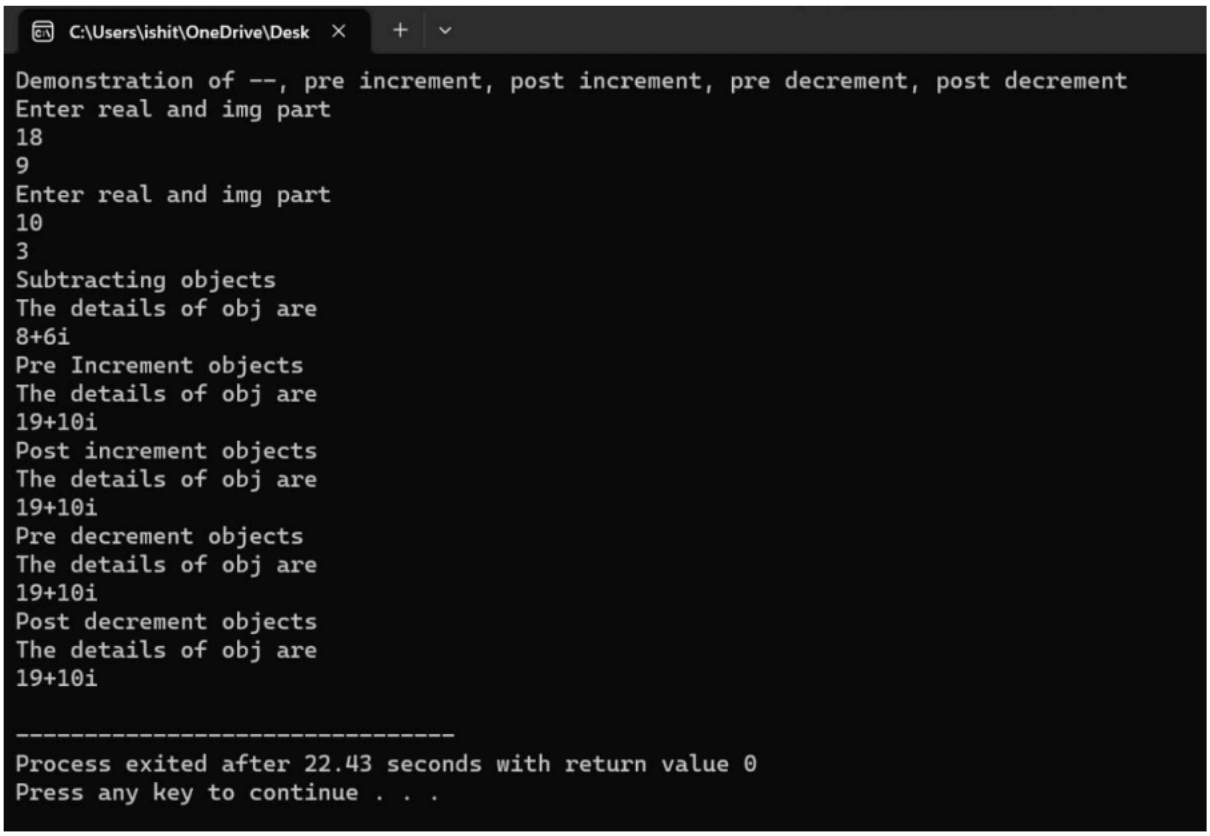
a. **Unary –**
b. **Unary ++ preincrement, postincrement**
c. **Unary -- predecrement, postdecrement**

```
#include<iostream>
using namespace std;
class complex
{
    int real,img;
    public:
    void input();
    void disp();
    complex operator-(complex obj2);
    complex operator++();
    complex operator++(int x);
    complex operator--();
    complex operator--(int x);
};
void complex::input()
{
    cout<<"Enter real and img part\n";
    cin>>real>>img;
}
void complex::disp()
{
    cout<<"The details of obj are\n";
    if(img<0)
    {
      cout<<real<<"-"<<-img<<"i"<<endl;
   }
   else
   {
      cout<<real<<"+"<<img<<"i"<<endl;
   }
}
complex complex::operator-(complex obj2)
{
    complex obj3;
    obj3.real=real-obj2.real;
    obj3.img=img-obj2.img;
    return obj3;
}
complex complex::operator++()
{
```

```
    real++;
    img++;
    return *this;
}
complex complex::operator++(int x)
{
    complex temp;
    temp.real= real;
    temp.img=img;
    real++;
    img++;
    return temp;
}
complex complex::operator--()
{
    real--;
    img--;
    return *this;
}
complex complex::operator--(int x)
{
    complex temp;
    temp.real=real;
    temp.img=img;
    real--;
    img--;
    return temp;
}
int main()
{
    complex obj1, obj2, obj3;
    cout<<"Demonstration of --, pre increment, post increment,
pre decrement, post decrement\n";
    obj1.input();
    obj2.input();
    cout<<"Subtracting objects\n";
    obj3=obj1-obj2;
    obj3.disp();
    cout<<"Pre Increment objects\n";
    obj3=++obj1;
    obj3.disp();
    cout<<"Post increment objects\n";
    obj3=obj1++;
    obj3.disp();
    cout<<"Pre decrement objects\n";
    obj3=--obj1;
```

```
    obj3.disp();
    cout<<"Post decrement objects\n";
    obj3=obj1--;
    obj3.disp();
    return 0;
}
```

Output:



15.Using the concept of operator overloading. Implement a program to overload the following:

With the help of friend function

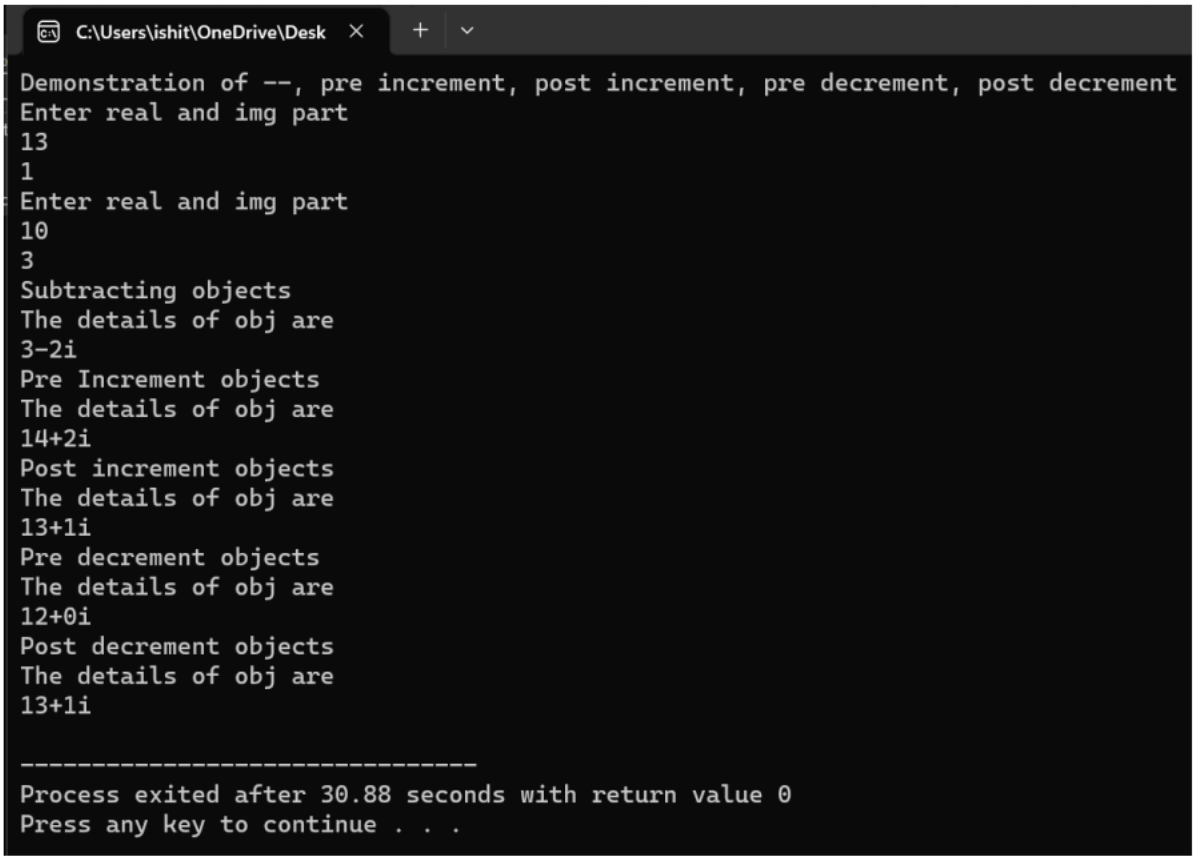
a. **Unary –**
b. **Unary ++ preincrement, postincrement**
c. **Unary -- predecrement, postdecrement**

```
#include<iostream>
using namespace std;
class complex
{
    int real,img;
    public:
    void input();
    void disp();
    friend complex operator-(complex obj1, complex obj2);
    friend complex operator++(complex obj1);
    friend complex operator++(complex obj1, int x);
    friend complex operator--(complex obj1);
    friend complex operator--(complex obj1, int x);
};
void complex::input()
{
    cout<<"Enter real and img part\n";
    cin>>real>>img;
}
void complex::disp()
{
    cout<<"The details of obj are\n";
    if(img<0)
    {
      cout<<real<<"-"<<-img<<"i"<<endl;
   }
   else
   {
      cout<<real<<"+"<<img<<"i"<<endl;
   }
    //cout<<"real : "<<real<<" img: "<<img<<endl;
}
complex operator-(complex obj1, complex obj2)
{
    complex obj3;
    obj3.real=obj1.real-obj2.real;
    obj3.img=obj1.img-obj2.img;
    return obj3;
}
```

```
complex operator++(complex obj1)
{
    obj1.real++;
    obj1.img++;
    return obj1;
}
complex operator++(complex obj1, int x)
{
    complex temp;
    temp.real= obj1.real;
    temp.img=obj1.img;
    obj1.real++;
    obj1.img++;
    return temp;
}
complex operator--(complex obj1)
{
    obj1.real--;
    obj1.img--;
    return obj1;
}
complex operator--(complex obj1, int x)
{
    complex temp;
    temp.real=obj1.real;
    temp.img=obj1.img;
    obj1.real--;
    obj1.img--;
    return temp;
}
int main()
{
    complex obj1, obj2, obj3;
    cout<<"Demonstration of --, pre increment, post increment,
pre decrement, post decrement\n";
    obj1.input();
    obj2.input();
    cout<<"Subtracting objects\n";
    obj3=obj1-obj2;
    obj3.disp();
    cout<<"Pre Increment objects\n";
    obj3=++obj1;
    obj3.disp();
    cout<<"Post increment objects\n";
    obj3=obj1++;
    obj3.disp();
```

```
    cout<<"Pre decrement objects\n";
    obj3=--obj1;
    obj3.disp();
    cout<<"Post decrement objects\n";
    obj3=obj1--;
    obj3.disp();
    return 0;
}
```

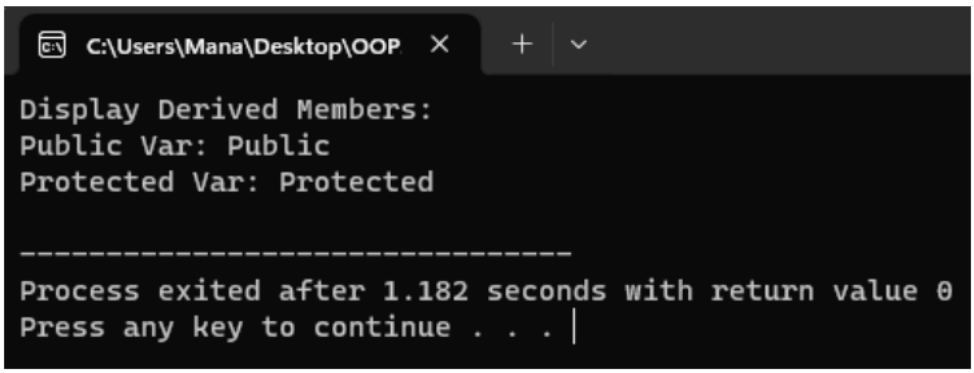
Output:



16.Create a Base class that consists of private, protected and public data members and member functions. Try using different access modifiers for inheriting Base class to the Derived class and create a table that summarizes the above three modes (when derived in public, protected and private modes) and shows the access specifier of the members of base class in the Derived class .

```
#include <iostream>
#include <string>
class Base {
/*private:
    std::string privateVar = "Private";*/
protected:
    std::string protectedVar = "Protected";
public:
    std::string publicVar = "Public";

    void display() {
        std::cout << "Display Base Members:" << std::endl;
        std::cout << "Public Var: " << publicVar << std::endl;
        std::cout << "Protected Var: " << protectedVar <<
std::endl;
        //std::cout << "Private Var: " << privateVar << std::endl;
    }
};

class Derived : public Base {
public:
    void displayMembers() {
        std::cout << "Display Derived Members:" << std::endl;
        std::cout << "Public Var: " << publicVar << std::endl;
        std::cout << "Protected Var: " << protectedVar <<
std::endl;
      //std::cout << "Private Var: " << privateVar << std::endl;
// Will generate an error
    }
};

int main() {
    Derived derived;
    derived.displayMembers();
    return 0;
}
```

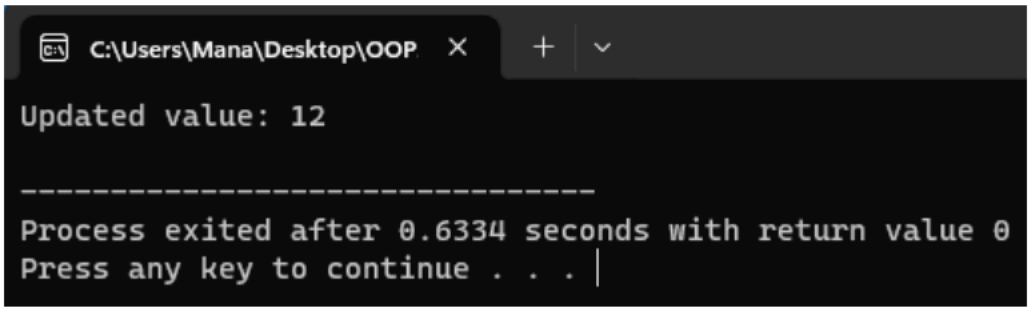
Output:



18. You are given three classes A, B and C. All three classes implement their own version of func. In class A, func multiplies the value passed as a parameter by 2. In class B, func multiplies the value passed as a parameter by 3. In class C, func multiplies the value passed as a parameter by 5.You are given class D such that You need to modify the class D and implement the function update_val which sets D's val to new_val by manipulating the value by only calling the func defined in classes A, B and C.It is guaranteed that new_val has only 2, 3 and 5 as its prime factors. Implement class D's function update_val. This function should update D's val only by calling A, B and C's func.

```
#include <iostream>
class A {
public:
    int func(int value) {
        return value * 2;
    }
};
class B {
protected:
    int func(int value) {
        return value * 3;
    }
};
/*class C {
private:
    int func(int value) {
        return value * 5;
    }
};*/
class D : public A, protected B{ //,private C
public:
    void update_val(int new_val) {
        int val = new_val;
        val = A::func(val);
        val = B::func(val);
       // val = C::func(val);
        std::cout << "Updated value: " << val << std::endl;
    }
};
int main() {
    D d;
    d.update_val(2);
    return 0;
}
```

Output:

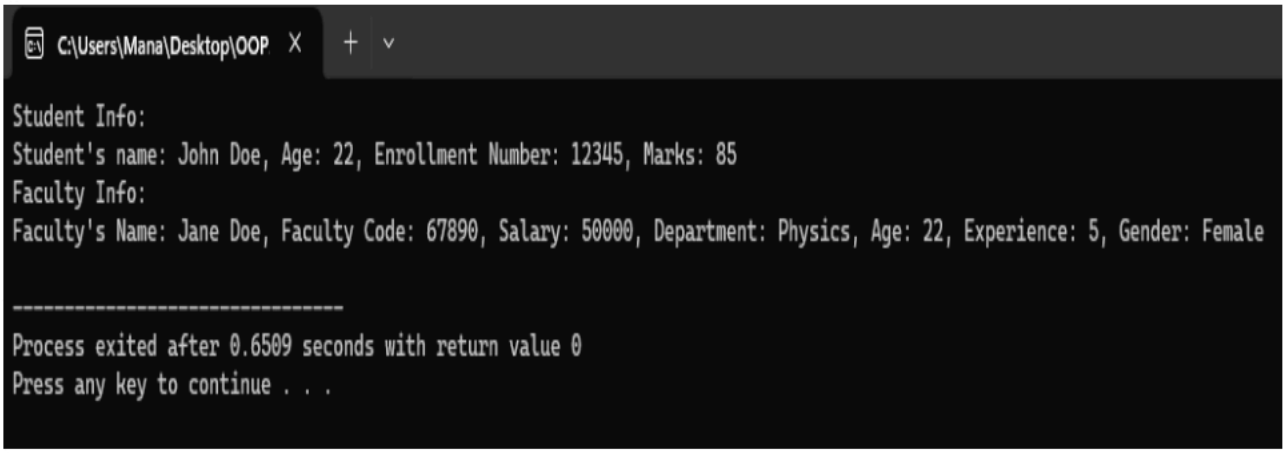


19. Create a class called Student that contains the data members like age, name, enroll_no, marks. Create another class called Faculty that contains data members like facultyName, facultyCode, salary,deptt, age, experience, gender. Create the function display() in both the classes to display the respective information. The derived Class Person demonstrates multiple inheritance. The program should be able to call both the base classes and displays their information. Remove the ambiguity (When we have exactly same variables or same methods in both the base classes, which one will becalled?) by proper mechanism.

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

class Student {
protected:
    int age;
    string name;
    int enroll_no;
    int marks;

public:
    Student(int a, string n, int e, int m) {
        age = a;
        name = n;
        enroll_no = e;
        marks = m;
    }

    void display() {
        cout << "Student's name: " << name << ", Age: " << age
             << ", Enrollment Number: " << enroll_no << ", Marks:
" << marks << endl;
    }
};

class Faculty {
protected:
    string facultyName;
    int facultyCode;
    int salary;
    string deptt;
    int age;
    int experience;
    string gender;

public:
    Faculty(string fn, int fc, int s, string d, int a, int e,
string g) {
        facultyName = fn;
        facultyCode = fc;
        salary = s;
        deptt = d;
        age = a;
        experience = e;
        gender = g;
    }
```

```
    void display() {
        cout << "Faculty's Name: " << facultyName << ", Faculty
Code: " << facultyCode
             << ", Salary: " << salary << ", Department: " <<
deptt << ", Age: " << age
             << ", Experience: " << experience << ", Gender: " <<
gender << endl;
    }
};

class Person : public Student, public Faculty {
public:
    Person(int a, string n, int e, int m, string fn, int fc, int
s, string d, int exp, string g)
        : Student(a, n, e, m), Faculty(fn, fc, s, d, a, exp, g) {}

    void display() {
        cout << "Student Info: " << endl;
        Student::display();
        cout << "Faculty Info: " << endl;
        Faculty::display();
    }
};

int main() {
    Person p(22, "John Doe", 12345, 85, "Jane Doe", 67890, 50000,
"Physics", 5, "Female");
    p.display();

    return 0;
}
```

Output:

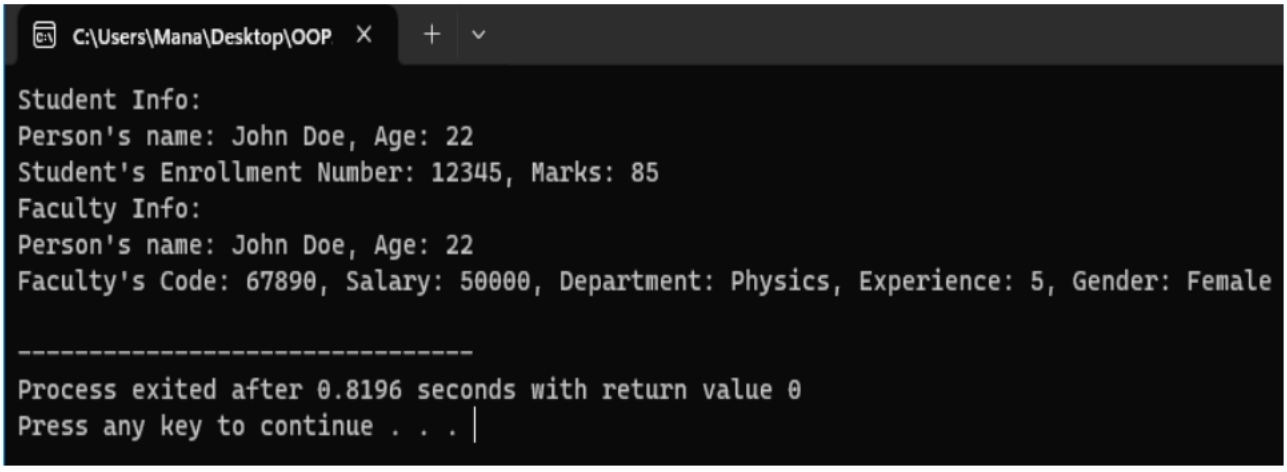


20. Implement a real case scenario by a proper C++ code to provide the solution to Diamond Problem in C++.

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

class Person {
protected:
    int age;
    string name;

public:
    Person(int a, string n) {
        age = a;
        name = n;
    }

    virtual void display() {
        cout << "Person's name: " << name << ", Age: " << age <<
endl;
    }
};

class Student : public virtual Person {
private:
    int enroll_no;
    int marks;

public:
    Student(int a, string n, int e, int m) : Person(a, n) {
        enroll_no = e;
        marks = m;
    }

    void display() {
        Person::display();
        cout << "Student's Enrollment Number: " << enroll_no << ",
Marks: " << marks << endl;
    }
};

class Faculty : public virtual Person {
private:
    int facultyCode;
    int salary;
    string deptt;
    int experience;
    string gender;

public:
    Faculty(int a, string n, int fc, int s, string d, int e,
string g) : Person(a, n) {
        facultyCode = fc;
        salary = s;
```

```
        deptt = d;
        experience = e;
        gender = g;
    }

    void display() {
        Person::display();
        cout << "Faculty's Code: " << facultyCode << ", Salary: "
<< salary << ", Department: " << deptt
             << ", Experience: " << experience << ", Gender: " <<
gender << endl;
    }
};

class PersonWithDetails : public Student, public Faculty {
public:
    PersonWithDetails(int a, string n, int e, int m, int fc, int
s, string d, int exp, string g)
        : Person(a, n), Student(a, n, e, m), Faculty(a, n, fc, s,
d, exp, g) {}

    void display() {
        cout << "Student Info: " << endl;
        Student::display();
        cout << "Faculty Info: " << endl;
        Faculty::display();
    }
};

int main() {
    PersonWithDetails p(22, "John Doe", 12345, 85, 67890, 50000,
"Physics", 5, "Female");
    p.display();

    return 0;
}
```

Output:

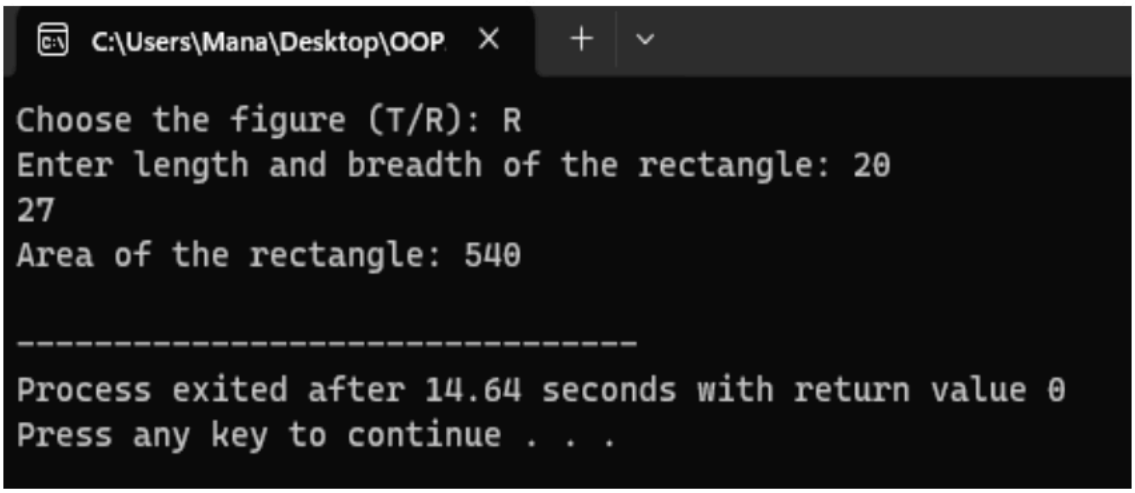


21. Create a base class called shape. Use this class to store two double type values that could be used to compute the area of figures. Derive two specific classes called triangle and rectangle from base shape. Add to the base class, a member function get_data() to initialize base class data members and another member function display_area() to compute and display the area of figures. Make display_area() as a virtual function and redefine this function in the derived class to suit their requirements. Using these three classes, design a program that will accept dimensions of a triangle or a rectangle interactively and display the area. Remember the two values given as input will be treated as lengths of two sides in the case of rectangles and as base and height in the case of triangle and used as follows:
Area of rectangle = x * y
Area of triangle = ½ *x*y

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

class Shape {
protected:
    double a, b;

public:
    virtual void get_data() {
        cout << "Enter two numbers: ";
        cin >> a >> b;
    }

    virtual void display_area() = 0;
};

class Triangle : public Shape {
public:
    void get_data() override {
        cout << "Enter base and height of the triangle: ";
        cin >> a >> b;
    }

    void display_area() override {
        cout << "Area of the triangle: " << 0.5 * a * b << endl;
    }
};

class Rectangle : public Shape {
public:
    void get_data() override {
        cout << "Enter length and breadth of the rectangle: ";
        cin >> a >> b;
    }

    void display_area() override {
        cout << "Area of the rectangle: " << a * b << endl;
    }
};

int main() {
    char choice;
```

```
    Shape* shape;

    cout << "Choose the figure (T/R): ";
    cin >> choice;

    if (choice == 'T' || choice == 't') {
        shape = new Triangle;
    } else if (choice == 'R' || choice == 'r') {
        shape = new Rectangle;
    } else {
        cout << "Invalid choice. Exiting...";
        return 0;
    }

    shape->get_data();
    shape->display_area();

    delete shape;

    return 0;
}
```

Output:

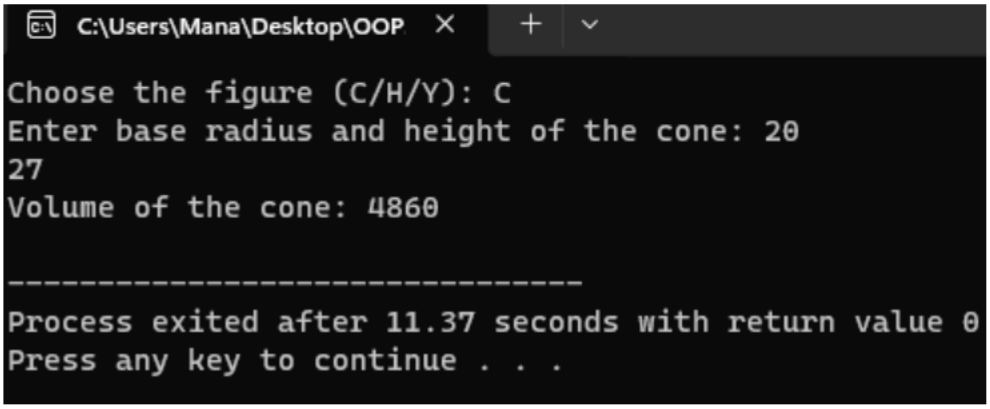


22. Create a base class called CAL_AREA(Abstract). Use this class to store float type values that could be used to compute the volume of figures. Derive two specific classes called cone, hemisphere and cylinder from the base CAL_AREA. Add to the base class, a member function getdata ( ) to initialize base class data members and another member function display volume( ) to compute and display the volume of figures. Make display volume ( ) as a pure virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes, design a program that will accept dimensions of a cone, cylinder and hemisphere interactively and display the volumes. Remember values given as input will be and used as follows:
Volume of cone = (1/3)pr2h
Volume of hemisphere = (2/3)pr3
Volume of cylinder = pr2h

```
#include <iostream>
#include <cmath>

class CAL_AREA {
protected:
    float a, b, c;

public:
    virtual void getdata() = 0;

    virtual void display_volume() = 0;
};

class CONE : public CAL_AREA {
public:
    void getdata() {
        std::cout << "Enter base radius and height of the cone: ";
        std::cin >> a >> b;
        c = b;
    }

    void display_volume() {
        std::cout << "Volume of the cone: " << (1.0/3.0) * a * b *
c << std::endl;
    }
};

class HEMISPHERE : public CAL_AREA {
public:
    void getdata() {
        std::cout << "Enter the radius of the hemisphere: ";
        std::cin >> a;
        b = c = a;
    }

    void display_volume() {
        std::cout << "Volume of the hemisphere: " << (2.0/3.0) * a
* b * c << std::endl;
    }
};
```

```
class CYLINDER : public CAL_AREA {
public:
    void getdata() {
        std::cout << "Enter the radius and height of the cylinder:
";
        std::cin >> a >> b;
        c = b;
    }

    void display_volume() {
        std::cout << "Volume of the cylinder: " << a * b * c <<
std::endl;
    }
};

int main() {
    char choice;
    CAL_AREA* cal_area;

    std::cout << "Choose the figure (C/H/Y): ";
    std::cin >> choice;

    if (choice == 'C' || choice == 'c') {
        cal_area = new CONE;
    } else if (choice == 'H' || choice == 'h') {
        cal_area = new HEMISPHERE;
    } else if (choice == 'Y' || choice == 'y') {
        cal_area = new CYLINDER;
    } else {
        std::cout << "Invalid choice. Exiting...";
        return 0;
    }

    cal_area->getdata();
    cal_area->display_volume();

    delete cal_area;

    return 0;
}
```

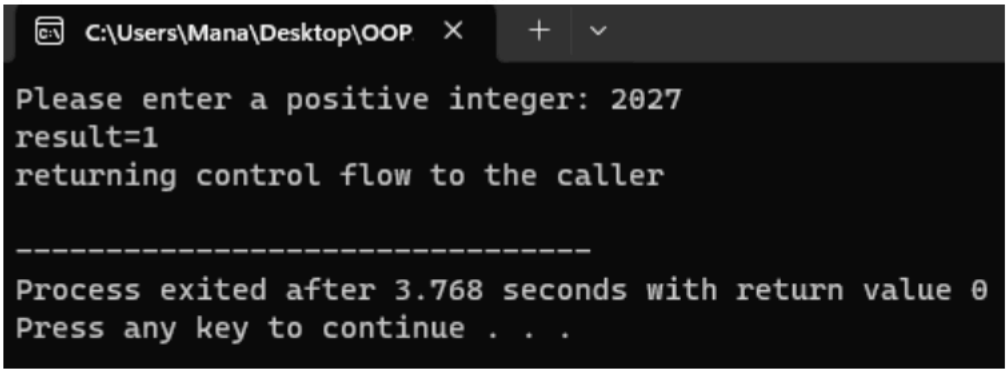
Output:



23. The task is to debug the existing code to successfully execute all provided test files. You are required to extend the existing code so that it handles the std::invalid_argument exception properly. More specifically, you have to extend the implementation of the process_input function. It takes integer n as an argument and has to work as follows:
1.It calls function largest_proper_divisor(n).
2.If this call returns a value without raising an exception, it should print in a single line result=d where d is the returned value.
3.Otherwise, if the call raises an invalid_argument exception, it has to print in a single line the string representation of the raised exception, i.e., its message.
4.Finally, no matter if the exception is raised or not, it should print in a single line returning control flow to the caller after any other previously printed output.

```
#include <iostream>
#include <stdexcept>
int largest_proper_divisor(int n) {
    if (n < 1) {
        throw std::invalid_argument("n must be a positive
integer");
    }
    for (int i = n / 2; i >= 1; i--) {
        if (n % i == 0) {
            return i;
        }
    }
    return n;
}
void process_input(int n) {
    try {
        int d = largest_proper_divisor(n);
        std::cout << "result=" << d << std::endl;} catch (const
std::invalid_argument& e) {
        std::cout << e.what() << std::endl;}
    std::cout << "returning control flow to the caller" <<
std::endl;
}
int main() {
    int n;
    std::cout << "Please enter a positive integer: ";
    std::cin >> n;
    process_input(n);
    return 0;
}
```

Output:

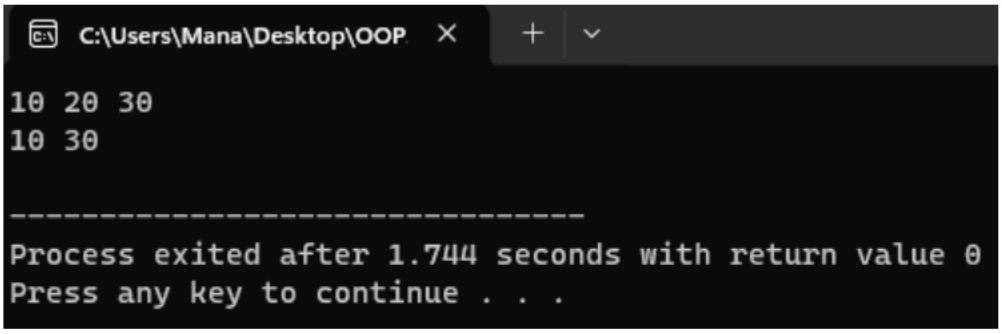


24. Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type. Write a program that can create a list (create a class list) of given type (int, float, char etc.) and perform insertion and deletion on list object.

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

template<typename T>
class Node {
public:
    T data;
    Node* next;

    Node(T data) {
        this->data = data;
        this->next = NULL; // Use NULL or 0 instead of nullptr
    }
};

template<typename T>
class List {
public:
    Node<T>* head;

    List() {
        head = NULL; // Use NULL or 0 instead of nullptr
    }

    void insert(T data) {
        Node<T>* newNode = new Node<T>(data);

        if (head == NULL) { // Use NULL or 0 instead of nullptr
            head = newNode;
        } else {
            Node<T>* last = head;
            while (last->next != NULL) { // Use NULL or 0 instead
of nullptr
                last = last->next;
            }
            last->next = newNode;
        }
    }

    void remove(T data) {
        if (head == NULL) { // Use NULL or 0 instead of nullptr
            return;
        }

        if (head->data == data) {
            Node<T>* temp = head;
            head = head->next;
            delete temp;
        } else {
            Node<T>* current = head;
```

```
            while (current->next != NULL) { // Use NULL or 0
instead of nullptr
                if (current->next->data == data) {
                    Node<T>* temp = current->next;
                    current->next = current->next->next;
                    delete temp;
                    return;
                }
                current = current->next;
            }
        }
    }

    void display() {
        Node<T>* temp = head;
        while (temp != NULL) { // Use NULL or 0 instead of nullptr
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    List<int> list;
    list.insert(10);
    list.insert(20);
    list.insert(30);
    list.display();

    list.remove(20);
    list.display();

    return 0;
}
```

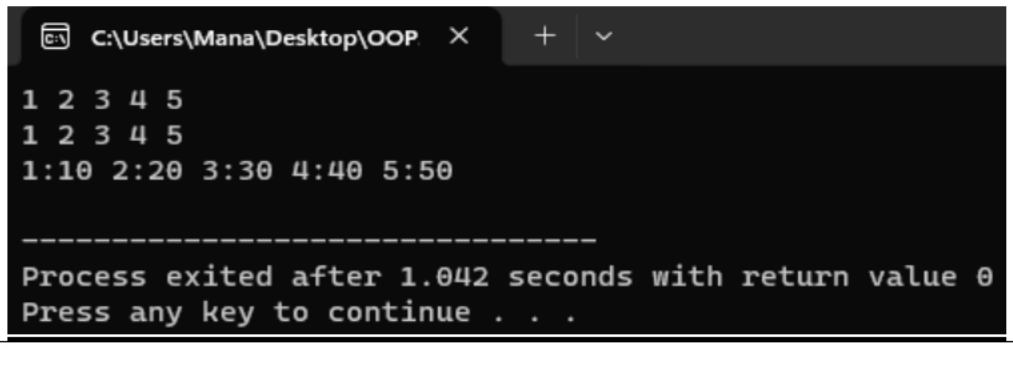
Output:



25. Construct a C++ program to demonstrate different methods of List, Vector and Map in STL (Standard Template Library).

```
#include <iostream>
#include <list>
#include <vector>
#include <map>
void displayList(std::list<int> lst) {
for (std::list<int>::iterator it= lst.begin();it!=lst.end(); it++)
    std::cout << *it << " ";
    std::cout << std::endl;
}
void displayVector(std::vector<int> vec) {
for(std::vector<int>::iterator it= vec.begin();it!=vec.end();it++)
    std::cout << *it << " ";
    std::cout << std::endl;
}
void displayMap(std::map<int, int> m) {
for (std::map<int,int>::iterator it= m.begin();it!= m.end();it++)
    std::cout << it->first << ":" << it->second << " ";
    std::cout << std::endl;
}
int main() {
    std::list<int> lst;
    lst.push_back(1);
    lst.push_back(2);
    lst.push_back(3);
    lst.push_back(4);
    lst.push_back(5);
    std::vector<int> vec;
    vec.push_back(1);
    vec.push_back(2);
    vec.push_back(3);
    vec.push_back(4);
    vec.push_back(5);
    std::map<int, int> m;
    m[1] = 10;
    m[2] = 20;
    m[3] = 30;
    m[4] = 40;
    m[5] = 50;
    displayList(lst);
    displayVector(vec);
    displayMap(m);
    return 0;
}
```

Output:



26. You are provided with a vector of N integers. Then, you are given 2 queries. For the first query, you are provided with 1 integer, which denotes a position in the vector. The value at this position in the vector needs to be erased. The next query consists of 2 integers denoting a range of the positions in the vector. The elements which fall under that range should be removed. The second query is performed on the updated vector which we get after performing the first query.

Input Format:

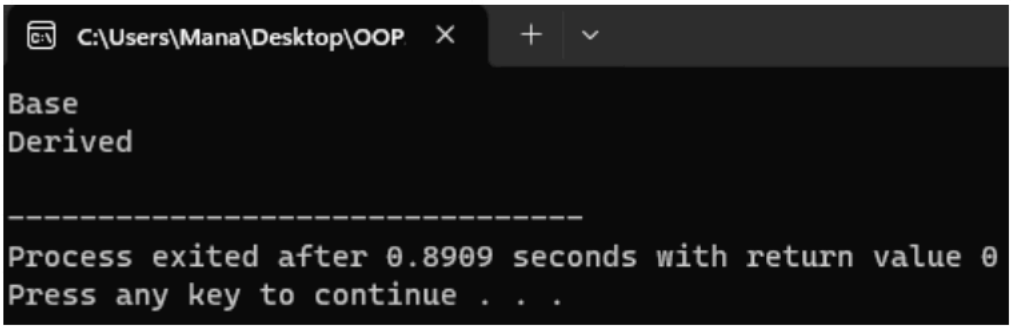
The first line of the input contains an integer N. The next line contains N space-separated integers (1-based index). The third line contains a single integer x, denoting the position of an element that should be removed from the vector. The fourth line contains two integers a and b denoting the range that should be erased from the vector inclusive of a and exclusive of b

Output Format:

Print the size of the vector in the first line and the elements of the vector after the two erase operations in the second line separated by space. Write a C++ Program to demonstrate the Concept of Dynamic binding with the help of virtual function.

```
#include<iostream>
#include<vector>

class Base {
public:
    virtual void show() { std::cout << "Base" << std::endl; }
};

class Derived : public Base {
public:
    void show() override { std::cout << "Derived" << std::endl; }
};

int main() {
    Base* basePtr = new Base();
    basePtr->show(); // Output: Base

    basePtr = new Derived();
    basePtr->show(); // Output: Derived

    return 0;
}
```

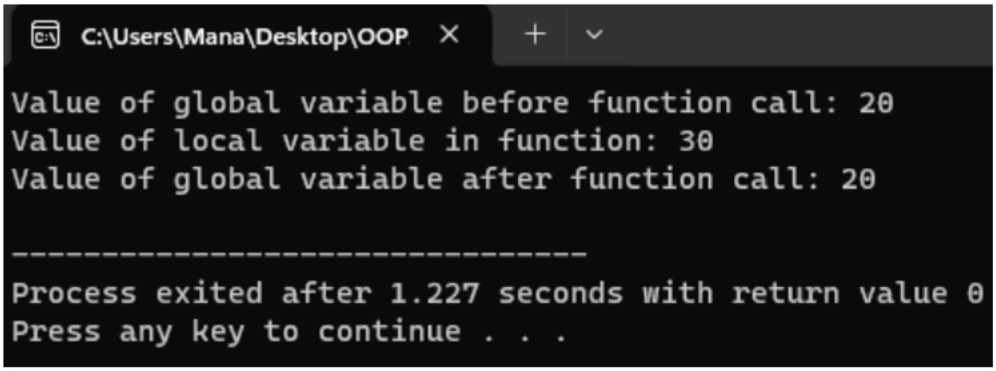
Output:



27. Write a C++ program to demonstrate the access of global variable when there is a local variable with same name.

```
#include<iostream>

int global_var = 20;

void function() {
    int global_var = 30;
    std::cout << "Value of local variable in function: " <<
global_var << std::endl;
}

int main() {
    std::cout << "Value of global variable before function call: "
<< global_var << std::endl;

    function();

    std::cout << "Value of global variable after function call: "
<< global_var << std::endl;

    return 0;
}
```

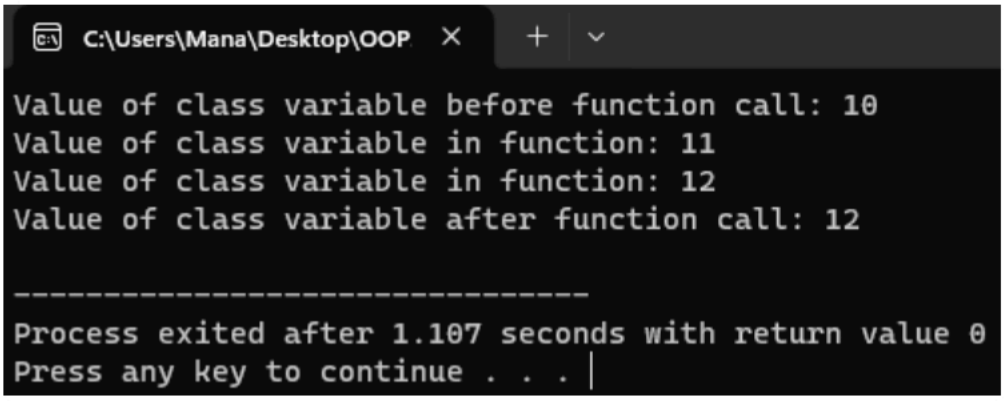
Output:



28. Write a C++ program to showcase the class objects as static.

```
#include<iostream>

class MyClass {
    public:
        static int classVar;
        void function() {
            classVar++;
            std::cout << "Value of class variable in function: "
<< classVar << std::endl;
        }
};

int MyClass::classVar = 10;

int main() {
    MyClass obj1;
    MyClass obj2;

    std::cout << "Value of class variable before function call: "
<< obj1.classVar << std::endl;

    obj1.function();
    obj2.function();

    std::cout << "Value of class variable after function call: "
<< obj1.classVar << std::endl;

    return 0;
}
```

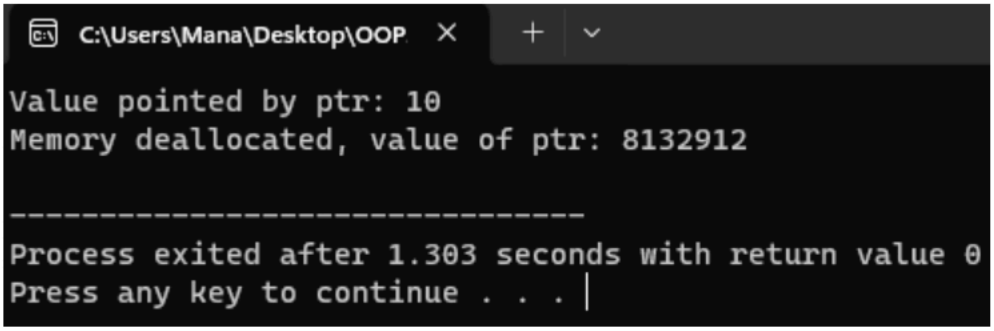
Output:



29. Write a C++ program to deallocate a memory pointed by pointer. Hint:(dangling pointer).

```
#include<iostream>

int main() {
    int *ptr = new int(10);
    std::cout << "Value pointed by ptr: " << *ptr << std::endl;

    delete ptr;
    std::cout << "Memory deallocated, value of ptr: " << *ptr <<
std::endl;

    return 0;
}
```

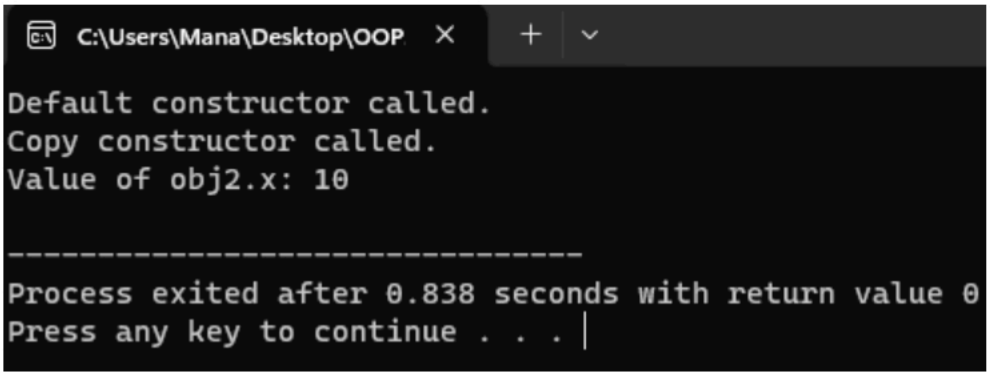
Output:



30. Write a C++ Program to copy the value of one object to another object using copy constructor.

```
#include<iostream>

class MyClass {
public:
    int x;

    // Default constructor
    MyClass() {
        std::cout << "Default constructor called." << std::endl;
    }

    // Copy constructor
    MyClass(const MyClass &source) {
        std::cout << "Copy constructor called." << std::endl;
        x = source.x;
    }
};

int main() {
    MyClass obj1;
    obj1.x = 10;

    MyClass obj2 = obj1;

    std::cout << "Value of obj2.x: " << obj2.x << std::endl;

    return 0;
}
```

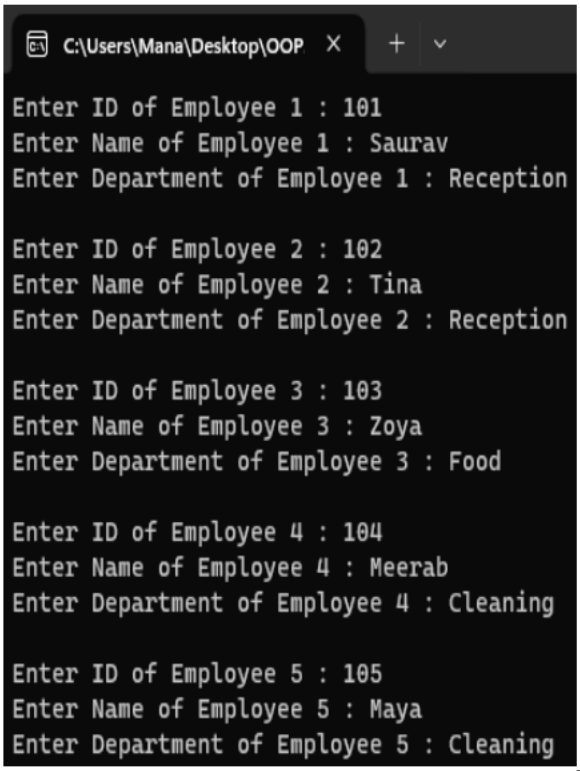
Output:

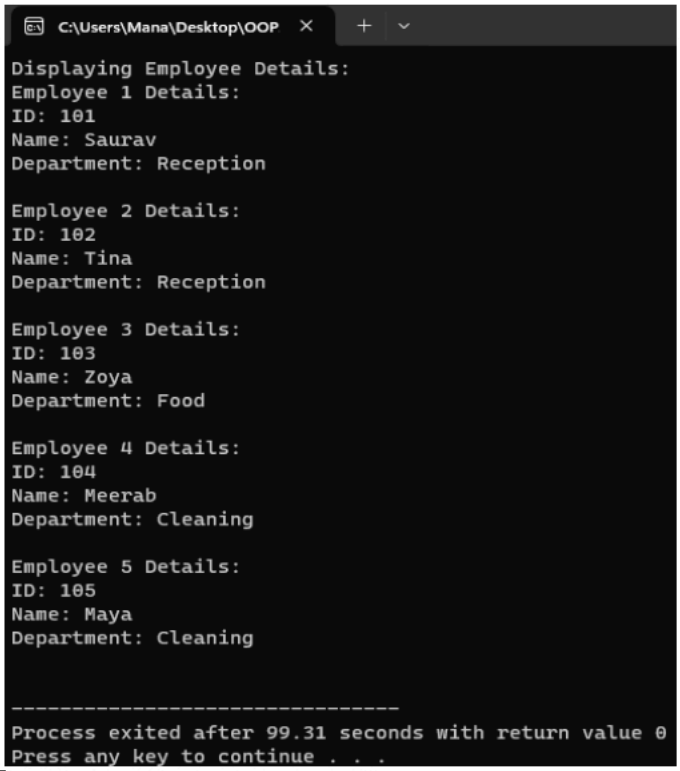


31. Write a C++ program to display Employee ID, Employee Name and Department who have joined the ABC Company. Declare the class of emp_id, emp_name, dept. Create an array of class objects. Read and Displays the Contents of the array.

```
#include<iostream>
#include<string>
using namespace std;
class Employee {
    public:
        int emp_id;
        string emp_name;
        string dept;
};
int main() {
    Employee employee[5];
    int i;
    for(i=0; i<5; i++) {
        cout << "Enter ID of Employee " << i+1 << " : ";
        cin >> employee[i].emp_id;
        cout << "Enter Name of Employee " << i+1 << " : ";
        cin >> employee[i].emp_name;
        cout << "Enter Department of Employee " << i+1 << " : ";
        cin >> employee[i].dept;
        cout << "\n"; }
    cout << "Displaying Employee Details: \n";
    for(i=0; i<5; i++) {
        cout << "Employee " << i+1 << " Details: \n";
        cout << "ID: " << employee[i].emp_id << "\n";
        cout << "Name: " << employee[i].emp_name << "\n";
        cout << "Department: " << employee[i].dept << "\n\n"; }
    return 0;}
```

Output:



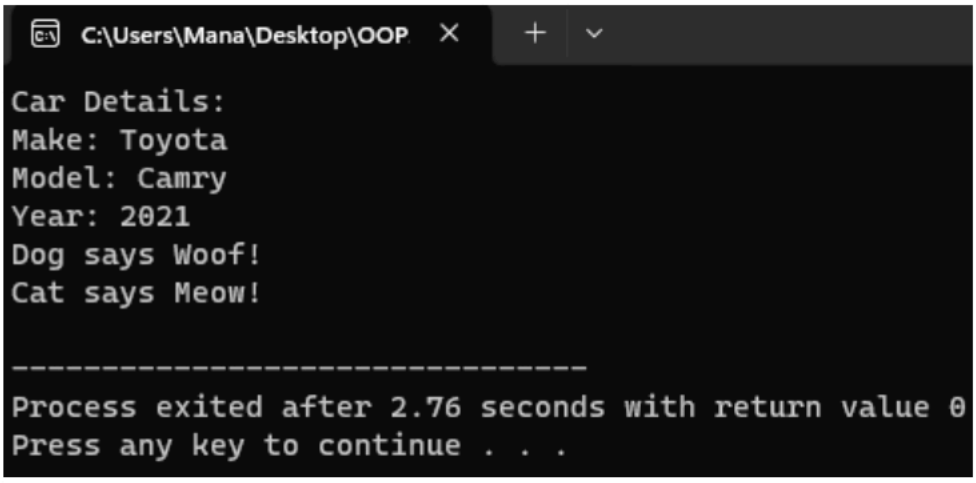


32. Write a C++ program to showcase the difference between Encapsulation and abstraction concept.

```
#include <iostream>
#include <string>

using namespace std;

// Encapsulation
class Car {
private:
    string make;
    string model;
    int year;

public:
    void setMake(string make) {
        this->make = make;
    }

    void setModel(string model) {
        this->model = model;
    }

    void setYear(int year) {
        this->year = year;
    }

    string getMake() {
        return make;
    }

    string getModel() {
        return model;
    }

    int getYear() {
        return year;
    }
};

// Abstraction
class Animal {
public:
    virtual void sound() = 0;
};

class Dog : public Animal {
public:
    void sound() {
        cout << "Dog says Woof!" << endl;
    }
};

class Cat : public Animal {
```

```
public:
    void sound() {
        cout << "Cat says Meow!" << endl;
    }
};

int main() {
    Car car;
    car.setMake("Toyota");
    car.setModel("Camry");
    car.setYear(2021);

    cout << "Car Details:" << endl;
    cout << "Make: " << car.getMake() << endl;
    cout << "Model: " << car.getModel() << endl;
    cout << "Year: " << car.getYear() << endl;

    Animal* animal;
    Dog dog;
    Cat cat;

    animal = &dog;
    animal->sound();

    animal = &cat;
    animal->sound();

    return 0;
}
```

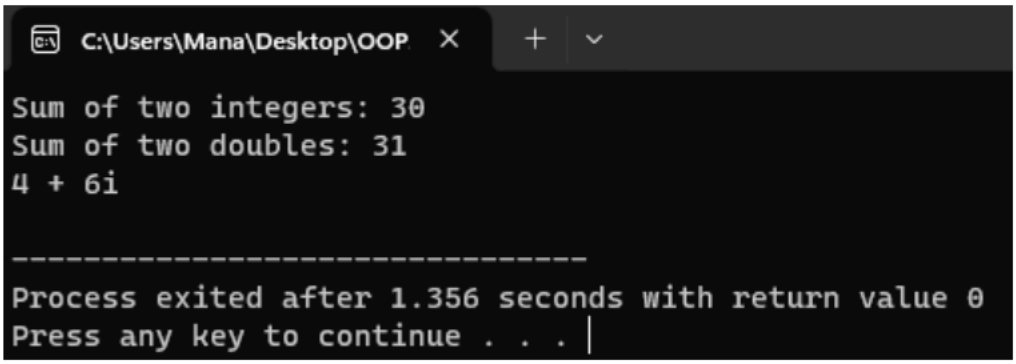
Output:



33. Write a C++ program to showcase o Function overloading o Operator overloading.

```
#include<iostream>
#include<cmath>

using namespace std;
// Function overloading
void addNumbers(int a, int b) {
    cout << "Sum of two integers: " << a + b << "\n";
}
void addNumbers(double a, double b) {
    cout << "Sum of two doubles: " << a + b << "\n";
}
// Operator overloading
class Complex {
    private:
        double real;
        double imaginary;
    public:
Complex(double r = 0.0, double i = 0.0) : real(r), imaginary(i) {}
        // Overloading the '+' operator
        Complex operator+(const Complex& c) {
           return Complex(real + c.real, imaginary + c.imaginary);
        }
        void print() {
            cout << real << " + " << imaginary << "i\n";
        }
};
int main() {
    // Function overloading
    addNumbers(10, 20);
    addNumbers(10.5, 20.5);
    // Operator overloading
    Complex c1(3.0, 4.0), c2(1.0, 2.0);
    Complex c3 = c1 + c2;
    c3.print();
    return 0;
}
```

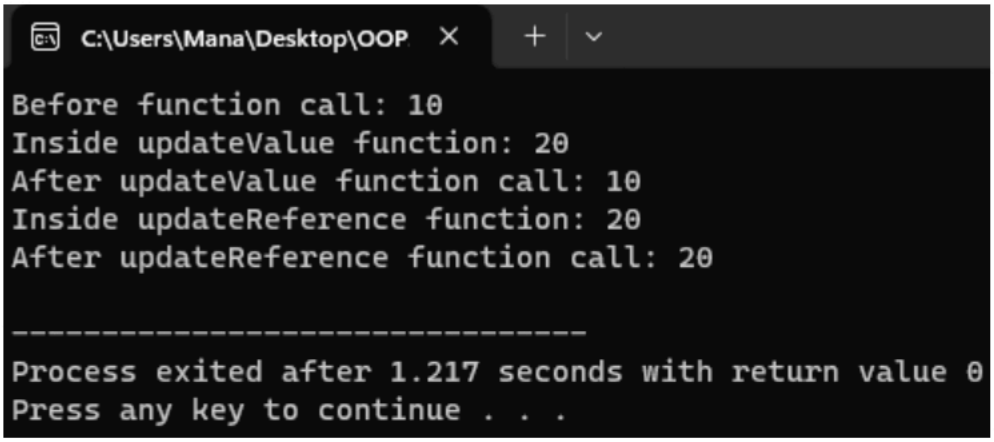
Output:



34. Write a C++ program to illustrate he difference between Call by reference and call by value.

```
#include <iostream>

using namespace std;

void updateValue(int value) {
    value = 20;
    cout << "Inside updateValue function: " << value << endl;
}

void updateReference(int &value) {
    value = 20;
    cout << "Inside updateReference function: " << value << endl;
}

int main() {
    int value = 10;
    cout << "Before function call: " << value << endl;

    updateValue(value);
    cout << "After updateValue function call: " << value << endl;

    updateReference(value);
    cout << "After updateReference function call: " << value <<
endl;

    return 0;
}
```

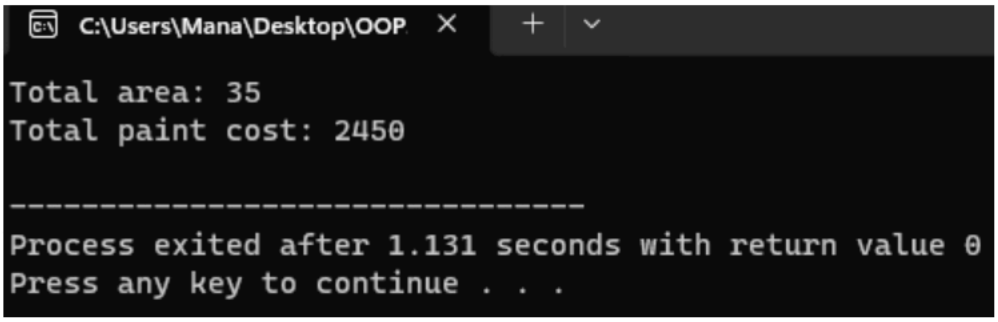
Output:



35. Write a C++ program to show multiple inheritance.

```
#include <iostream>
using namespace std;
class Shape {
public:
    void setWidth(int w) {
        width = w;        }
    void setHeight(int h) {
        height = h;        }
protected:
    int width;
    int height;
};
class PaintCost {
public:
    int getCost(int area) {
        return area * 70; }
};
class Rectangle: public Shape, public PaintCost {
public:
    int getArea() {
        return (width * height); }
};
int main() {
    Rectangle rect;
    int area;
    rect.setWidth(5);
    rect.setHeight(7);
    area = rect.getArea();
    cout << "Total area: " << area << endl;
    cout << "Total paint cost: " << rect.getCost(area) << endl;
    return 0;
}
```

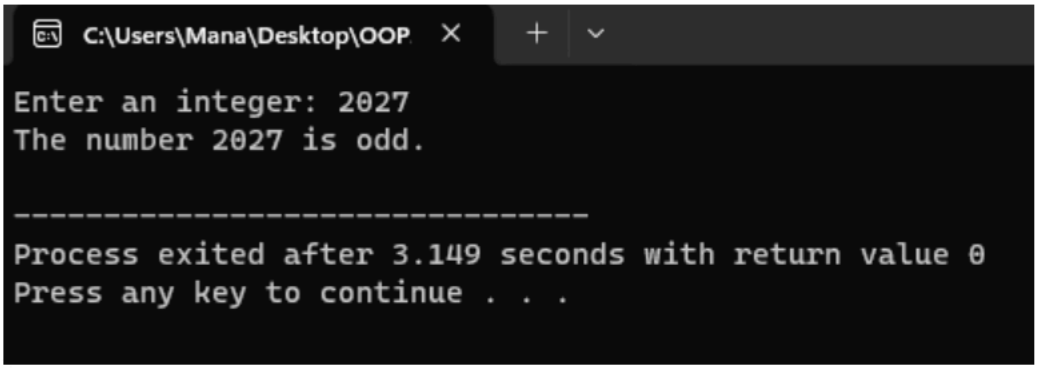
Output:



36. Any integer is input by the user. Write a program to find out whether it is an odd number or an even number .

```
#include <iostream>

using namespace std;

int main() {
    int num;
    cout << "Enter an integer: ";
    cin >> num;

    if (num % 2 == 0) {
        cout << "The number " << num << " is even." << endl;
    } else {
        cout << "The number " << num << " is odd." << endl;
    }

    return 0;
}
```

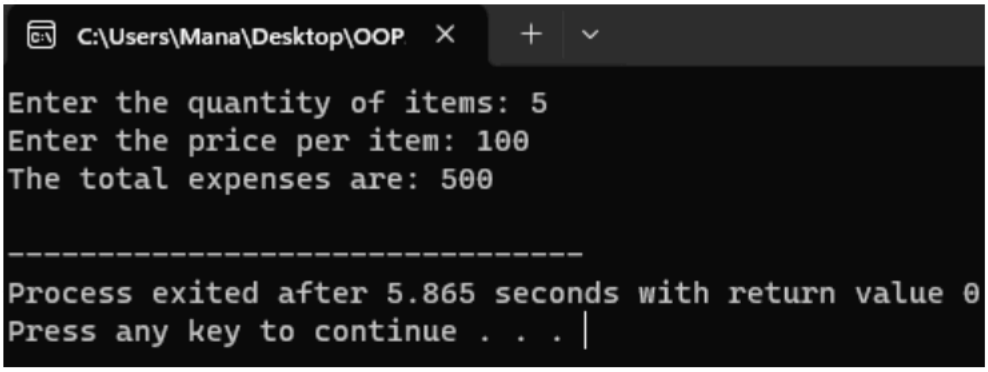
Output:



37. Write a program to calculate the total expenses. Quantity and price per item are input by the user and a discount of 10% is offered if the expense is more than 5000.

```
#include <iostream>

using namespace std;

int main() {
    int quantity;
    float price_per_item, total_expenses;

    cout << "Enter the quantity of items: ";
    cin >> quantity;

    cout << "Enter the price per item: ";
    cin >> price_per_item;

    total_expenses = quantity * price_per_item;

    if (total_expenses > 5000) {
        total_expenses -= total_expenses * 0.10;
    }

    cout << "The total expenses are: " << total_expenses << endl;

    return 0;
}
```

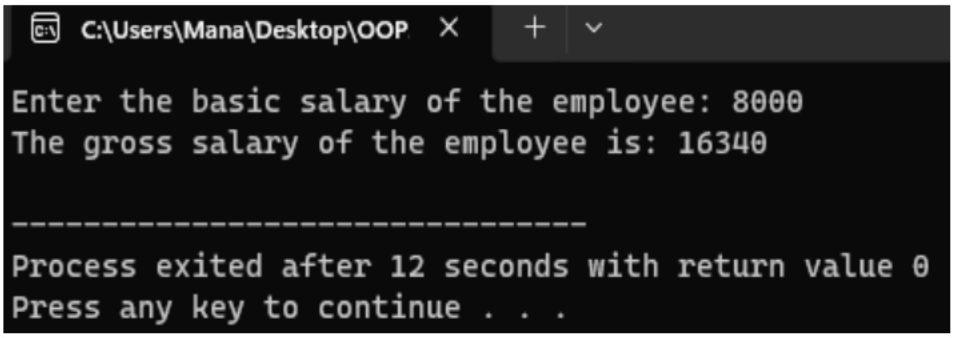
Output:



38. In a company an employee is paid as follows: If his basic salary is less than Rs. 1500, then HRA = 10% of the basic salary and DA = 90% of basic salary. If his salary is either equal to or above Rs. 1500, then HRA = Rs. 500and DA = 98% of basic salary. If the employee's salary is input by the user write a program to find his gross salary.

```
#include <iostream>

using namespace std;

int main() {
    float basic_salary;

    cout << "Enter the basic salary of the employee: ";
    cin >> basic_salary;

    float hra, da;

    if (basic_salary < 1500) {
        hra = 0.10 * basic_salary;
        da = 0.90 * basic_salary;
    } else {
        hra = 500;
        da = 0.98 * basic_salary;
    }

    float gross_salary = basic_salary + hra + da;

    cout << "The gross salary of the employee is: " <<
gross_salary << endl;

    return 0;
}
```

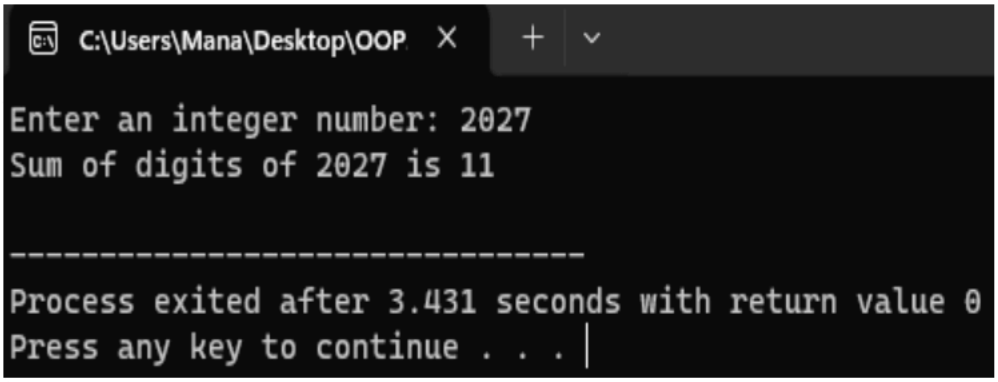
Output:



39. Write a program to sum the digits of a given integer number.

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

int sumDigits(int number) {
    int sum = 0;
    while (number > 0) {
        sum += number % 10;
        number /= 10;
    }
    return sum;
}

int main() {
    int number;
    cout << "Enter an integer number: ";
    cin >> number;

    cout << "Sum of digits of " << number << " is " <<
sumDigits(number) << endl;

    return 0;
}
```

Output:

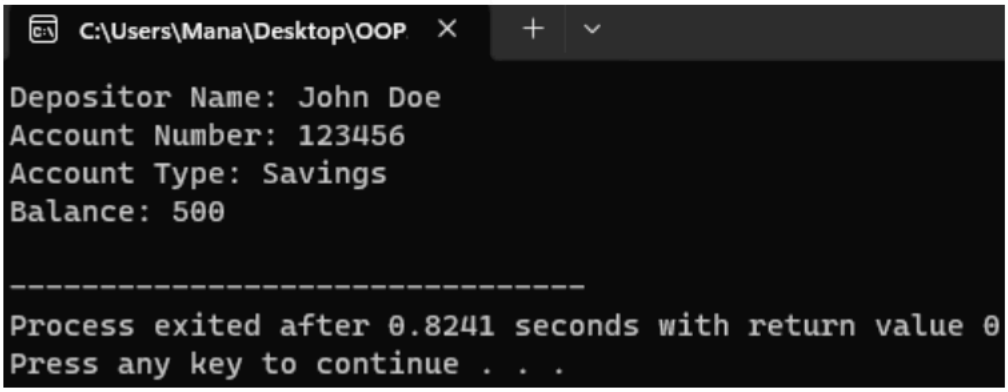


40. Define a class to represent a bank account. Include the following members: Data Members: 1) Name of depositor2) Account number3) Type of account4) Balance amount in the accountMember Functions: 1) to assign initial values2) To deposit an amount3) To withdraw amount after checking the balance.

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

class BankAccount {
    private:
        string depositorName;
        int accountNumber;
        string accountType;
        double balance;

    public:
        BankAccount() {
            depositorName = "";
            accountNumber = 0;
            accountType = "";
            balance = 0.0;
        }

        void setDetails(string depositorName, int accountNumber,
string accountType) {
            this->depositorName = depositorName;
            this->accountNumber = accountNumber;
            this->accountType = accountType;
        }

        void deposit(double amount) {
            balance += amount;
        }

        bool withdraw(double amount) {
            if (balance >= amount) {
                balance -= amount;
                return true;
            }
            else {
                cout << "Insufficient balance!" << endl;
                return false;
            }
        }

        void displayBalance() {
            cout << "Depositor Name: " << depositorName << endl;
            cout << "Account Number: " << accountNumber << endl;
            cout << "Account Type: " << accountType << endl;
            cout << "Balance: " << balance << endl;
        }
};

int main() {
    BankAccount account;
```

```
    account.setDetails("John Doe", 123456, "Savings");
    account.deposit(1000.0);
    account.withdraw(500.0);

    account.displayBalance();

    return 0;
}
```

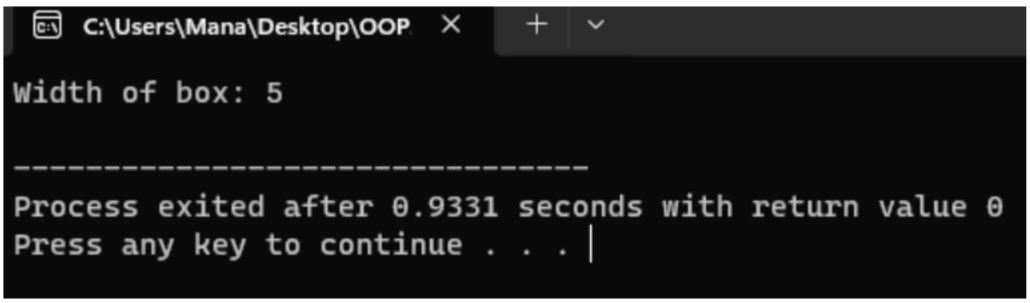
Output:



42.Write a class named box with data member width of the box and member functions width to set the width of the box and a non-member function printwidth to print the width of box. Printwidth function is the friend of the setwidth function. Use necessary objects and access them.

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

class box {
    int width;
public:
    // member function to set the width of the box
    void setwidth(int w) {
        width = w;
    }

    // friend function to print the width of the box
    friend void printwidth(box box1);
};

void printwidth(box box1) {
    cout << "Width of box: " << box1.width << endl;
}

int main() {
    box box1;

    // setting the width of the box
    box1.setwidth(5);

    // printing the width of the box
    printwidth(box1);

    return 0;
}
```

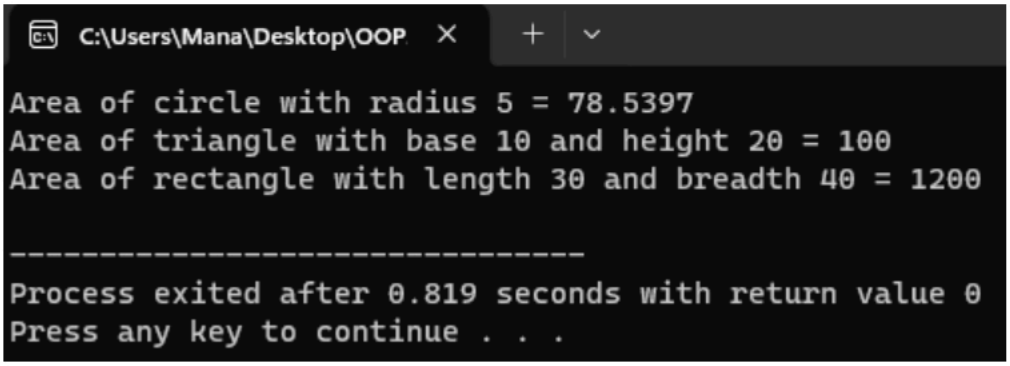
Output:



43. Write a C++ program to find area of a circle, triangle and rectangle. Use function overloading concept.

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

// Function to calculate the area of a circle
double calculateCircleArea(double radius) {
    return 3.14159 * pow(radius, 2);
}

// Function to calculate the area of a triangle
double calculateTriangleArea(double base, double height) {
    return 0.5 * base * height;
}

// Function to calculate the area of a rectangle
double calculateRectangleArea (double length, double breadth) {
    return length * breadth;
}

int main() {
    double radius = 5.0;
    double base = 10.0;
    double height = 20.0;
    double length = 30.0;
    double breadth = 40.0;

    cout << "Area of circle with radius " << radius << " = " <<
calculateCircleArea(radius) << endl;
    cout << "Area of triangle with base " << base << " and height
" << height << " = " << calculateTriangleArea(base, height) <<
endl;
    cout << "Area of rectangle with length " << length << " and
breadth " << breadth << " = " << calculateRectangleArea(length,
breadth) << endl;

    return 0;
}
```

Output:



44.Specify a class complex to represent the complex number and overload +,-,* and /operators when working on the objects of the class.

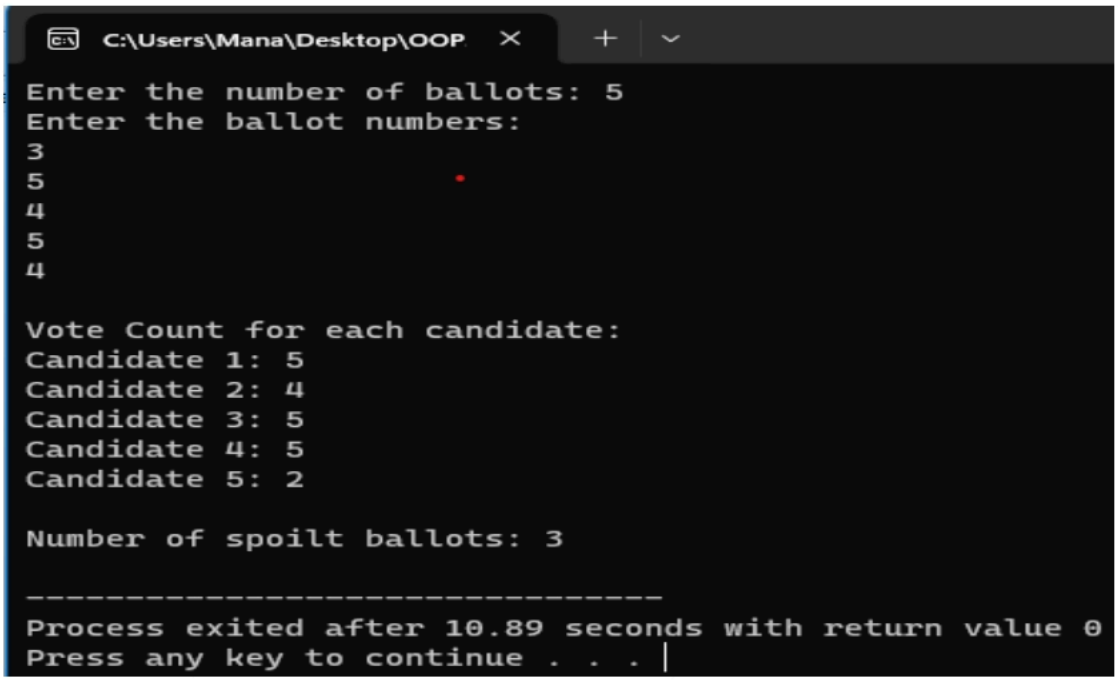
```
#include<iostream>
class complex {
    private:
        double real;
        double imag;
    public:
     complex(double r = 0.0, double i = 0.0) : real(r), imag(i){}
        // overloading + operator
        complex operator +(complex const &obj) {
            return complex(real + obj.real, imag + obj.imag);}
        // overloading - operator
        complex operator -(complex const &obj) {
            return complex(real - obj.real, imag - obj.imag);}
        // overloading * operator
        complex operator *(complex const &obj) {
            return complex(real * obj.real - imag * obj.imag,
                           real * obj.imag + imag * obj.real);}
        // overloading / operator
        complex operator /(complex const &obj) {
            double r = obj.real;
            double i = obj.imag;
            double denominator = r * r + i * i;
            return complex((real * r + imag * i) / denominator,
                           (imag * r - real * i) / denominator);}
        // print function
        void print() {
            std::cout << real << " + i" << imag << std::endl;}
};
int main() {
    complex c1(3.0, 2.0), c2(1.0, 1.0);
    std::cout << "c1 = "; c1.print();
    std::cout << "c2 = "; c2.print();
    complex c3 = c1 + c2;
    std::cout << "c1 + c2 = "; c3.print();
    complex c4 = c1 - c2;
    std::cout << "c1 - c2 = "; c4.print();
    complex c5 = c1 * c2;
    std::cout << "c1 * c2 = "; c5.print();
    complex c6 = c1 / c2;
    std::cout << "c1 / c2 = "; c6.print();
    return 0;}
```

Output:

45.An election is contested by five candidates. The candidates are numbered 1 to 5 and voting is done by marking the candidate number in a ballot paper. Write a C++ program to read the ballot and count the votes cast for each candidate using an array variable count. In case, a number read is outside the range 1 to 5 the ballot should be considered as a 'spoilt ballot', and the program should also count the number of spoilt ballot .

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

int main() {
    int n, i, count[6] = {0};

    cout << "Enter the number of ballots: ";
    cin >> n;

    cout << "Enter the ballot numbers:\n";
    for(i = 0; i < n; i++) {
        cin >> count[i];
        // increment the corresponding candidate count
        if(count[i] >= 1 && count[i] <= 5)
            count[count[i]]++;
        else
            count[0]++;    // increment the spoilt ballot count
    }
    cout << "\nVote Count for each candidate:\n";
    for(i = 1; i <= 5; i++)
        cout << "Candidate " << i << ": " << count[i] << endl;

    cout << "\nNumber of spoilt ballots: " << count[0] << endl;

    return 0;
}
```

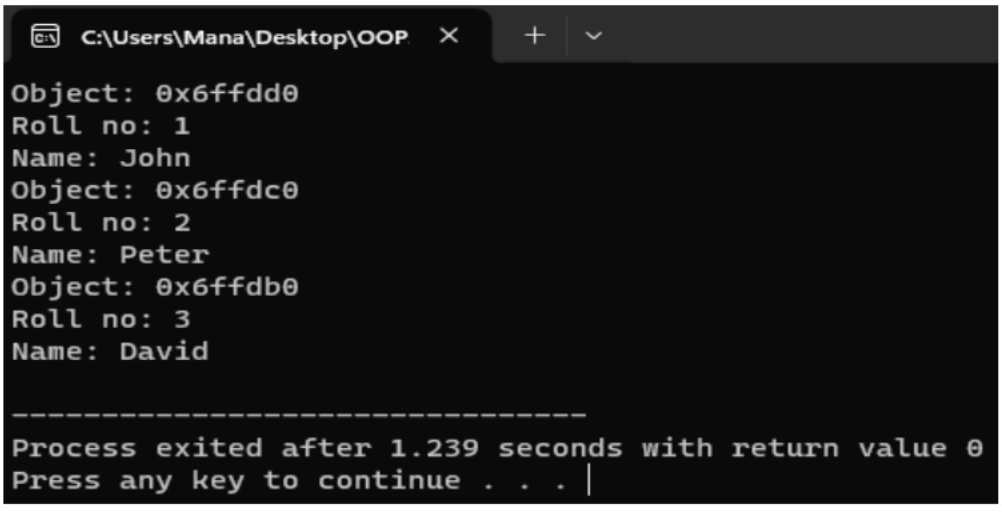
Output:



46.Write a C++ program to create three objects for a class named pntr_obj with data members such as roll_no & name . Create a member function set_data() for setting the data values and print()member function to print which object has invoked it using 'this' pointer.

```
#include<iostream>

class pntr_obj {
private:
    int roll_no;
    std::string name;

public:
    void set_data(int roll_no, std::string name) {
        this->roll_no = roll_no;
        this->name = name;
    }

    void print() {
        std::cout << "Object: " << this << std::endl;
        std::cout << "Roll no: " << roll_no << std::endl;
        std::cout << "Name: " << name << std::endl;
    }
};

int main() {
    pntr_obj obj1, obj2, obj3;

    obj1.set_data(1, "John");
    obj2.set_data(2, "Peter");
    obj3.set_data(3, "David");

    obj1.print();
    obj2.print();
    obj3.print();

    return 0;
}
```

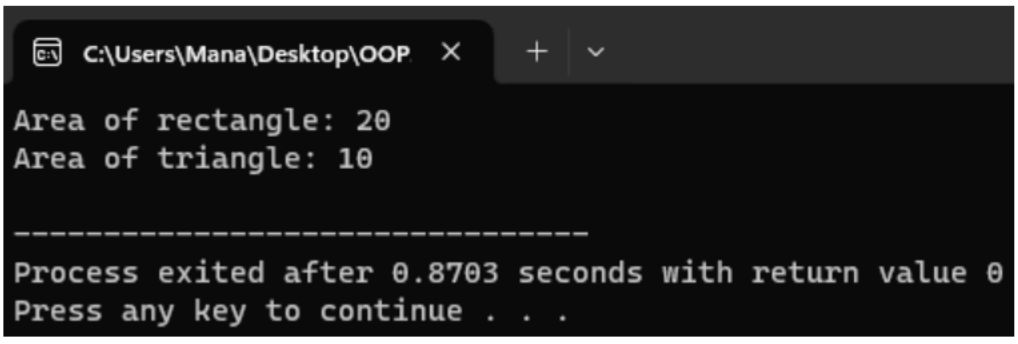
Output:



47.Write a C++ program to explain virtual function (polymorphism) by creating a base class_polygon which has virtual function area(). Two classes c_rectangle and c_traingle derived from c_polygon and they have area() to calculate and return the area of rectangle and triangle respectively.

```
#include<iostream>
using namespace std;
// Base class
class c_polygon {
public:
    virtual double area() {
        cout << "Base class area." << endl;
        return 0;}
};
// Derived class 1
class c_rectangle : public c_polygon {
    double length, breadth;
public:
    c_rectangle(double len, double bre) : length(len),
breadth(bre) {}
    double area() {
        cout << "Area of rectangle: " << length * breadth << endl;
        return length * breadth;}
};
// Derived class 2
class c_triangle : public c_polygon {
    double base, height;
public:
    c_triangle(double bas, double hei) : base(bas), height(hei) {}
    double area() {
        cout << "Area of triangle: " << 0.5 * base * height <<
endl;
        return 0.5 * base * height;}
};
int main() {
    c_polygon *p;
    c_rectangle r(4, 5);
    c_triangle t(4, 5);
    p = &r;
    p->area();
    p = &t;
    p->area();
    return 0;}
```

Output:



48. Write a program to accept the student details such as name and 3 different marks by the get_data() method and display the name and average of marks using the display() method. Define a friend class for calculating the average of marks using the method mark_avg().

```
#include<iostream>
class Student {
private:
    std::string name;
    int marks[3];
public:
    void get_data() {
        std::cout << "Enter name: ";
        std::cin >> name;
        std::cout << "Enter 3 marks: ";
        for(int i = 0; i < 3; i++) {
            std::cin >> marks[i];  }
    }
    void display() {
        float average = mark_avg();
        std::cout << "Name: " << name << std::endl;
        std::cout << "Average of marks: " << average << std::endl;
    }
    float mark_avg() {
        float sum = 0;
        for(int i = 0; i < 3; i++) {
            sum += marks[i];
        }
        return sum / 3;}
    friend class Test;
};
class Test {
public:
    void check() {
        Student s;
        s.get_data();
        s.display();}
};
int main() {
    Test t;
    t.check();
    return 0;}
```

Output:

```
C:\Users\Mana\Desktop\OOP
Enter name: Saurav
Enter 3 marks: 95
98
96
Name: Saurav
Average of marks: 96.3333

--------------------------------
Process exited after 14.09 seconds with return value 0
Press any key to continue . . .
```

49. Write a program to explain the class template by creating a template T for a class named pair having two data members of type T which are inputted by a constructor and a member function getmax() returns the greatest of two numbers to main. Note: the value of T depends upon the datatype specified during object creation.

```
#include<iostream>
// Creating a class template named 'pair'
template<typename T>
class pair {
private:
    T first, second;

public:
    // Constructor
    pair(T f, T s) {
        first = f;
        second = s;
    }

    // Member function to return the maximum number
    T get_max() {
        return (first > second) ? first : second;
    }
};
int main() {
    // Creating objects of pair class template with different data
types
    pair<int> intPair(10, 20);
    pair<float> floatPair(10.5, 20.5);
    pair<double> doublePair(10.7, 20.7);
    // Displaying the maximum number of each pair
    std::cout << "Maximum of integer pair: " << intPair.get_max()
<< std::endl;

    std::cout << "Maximum of float pair: " << floatPair.get_max()
<< std::endl;
    std::cout << "Maximum of double pair: " <<
doublePair.get_max() << std::endl;

    return 0;
}
```

Output: