**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**

**CHANDUBHAI S. PATEL INSTITUTE OF TECHNOLOGY**

U & P U. Patel Department of Computer Engineering

**F.Y. B.Tech (CE/IT/EC)**

**Subject Name:** Object Oriented Programming with C++ **Semester :** II

**Subject Code:** CE142 **Academic year:** 2016-17

**1. Write a C++ program that will output this passage by Deepak Chopra. Make sure your output looks exactly as shown here (including spacing, line breaks, punctuation, and the title and author). Use cout and cin objects and endl manipulator.**

**“You alone are the judge of your worth**

**and your goal is to discover infinite**

**worth in yourself, no matter what**

**anyone else thinks.” by Deepak Chopra**

**To include a quotation mark in your output, try this: \"**

#include<iostream>

#include<cstdio>

using namespace std;

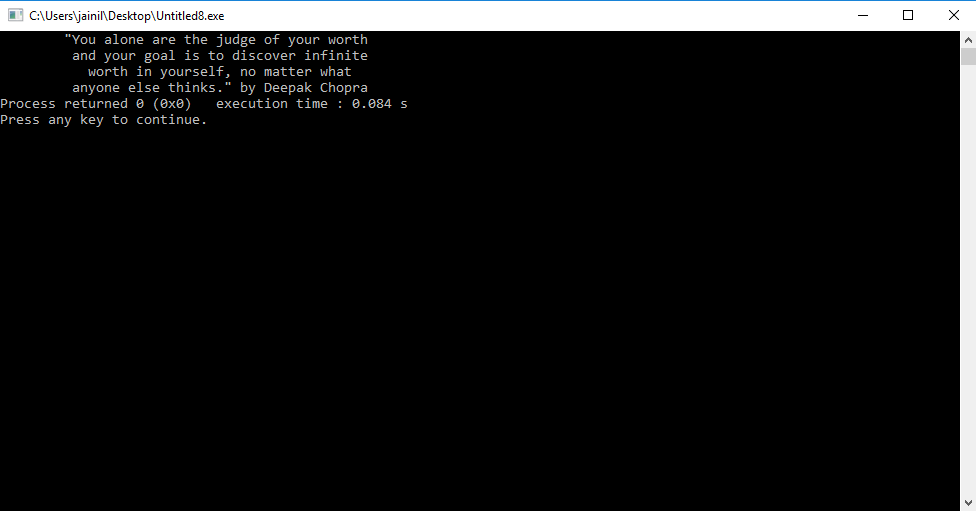
int main()

{

cout<<"\t\"You alone are the judge of your worth"<<endl<<"\t and your goal is to discover infinite"<<endl<<"\t worth in yourself, no matter what"<<endl<<"\t anyone else thinks.\" by Deepak Chopra";

return 0;

}



**2. write a program to create the following table. Use endl and setw manipulator.**

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** | **3** | **4** |
| **2** | **4** | **6** | **8** |
| **3** | **6** | **9** | **12** |
| **4** | **8** | **12** | **16** |

#include<iostream>

#include<iomanip>

using namespace std;

int main()

{

for(int i=1;i<=4;i++)

{

for(int j=1;j<=4;j++)

{

cout<<setw(4)<<i\*j;

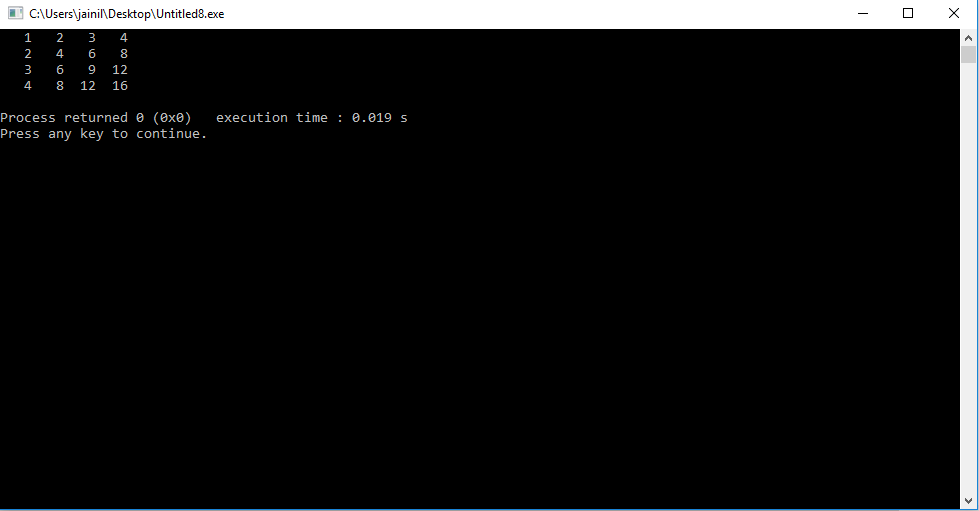
}

cout<<endl;

}

return 0;

}



**3. Write a C++ program to add two floating numbers using pointer. The result should contain only two digits after the decimal.**

**Use fixed, scientific and setprecision () manipulators for controlling the precision of floating point numbers.**

/\*Write a C++ program to add two floating numbers using pointer. The result should contain only two digits after the decimal.

Use fixed, scientific and setprecision () manipulators for controlling the precision of floating point numbers.\*/

#include<iostream>

#include<cstdio>

#include<iomanip>

using namespace std;

int main()

{

float a,b,c;

float \*p,\*q,\*r;

p=&a;

q=&b;

r=&c;

cout<<"enter first number"<<endl;

cin>>\*p;

cout<<"enter second number"<<endl;

cin>>\*q;

\*r=\*p+\*q;

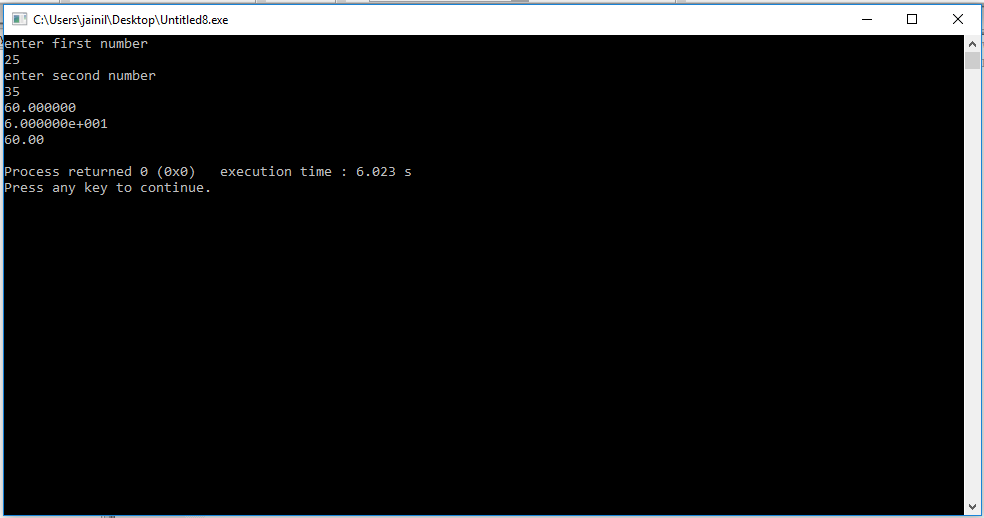
cout<<fixed<<\*r<<endl;

cout<<scientific<<\*r<<endl;

cout<<fixed<<setprecision(2)<<\*r<<endl;

return 0;

}



**4. Find error in the following code and give reasons for each error:**

**Can we declare an array of references? Can we assign NULL value to reference variable? Is Reference variable a pointer variable? Can we declare a reference variable without initializing it?**

#include<iostream>

using namespace std;

int main()

{

int a =10, b=12;

int & r;

int & c = NULL;

int & d[2] = {a,b};

cout<<"r = "<< r;

return 0;

}

PROGRAM:

#include<iostream>

using namespace std;

int main()

{

int a =10, b=12;

int &r=a;

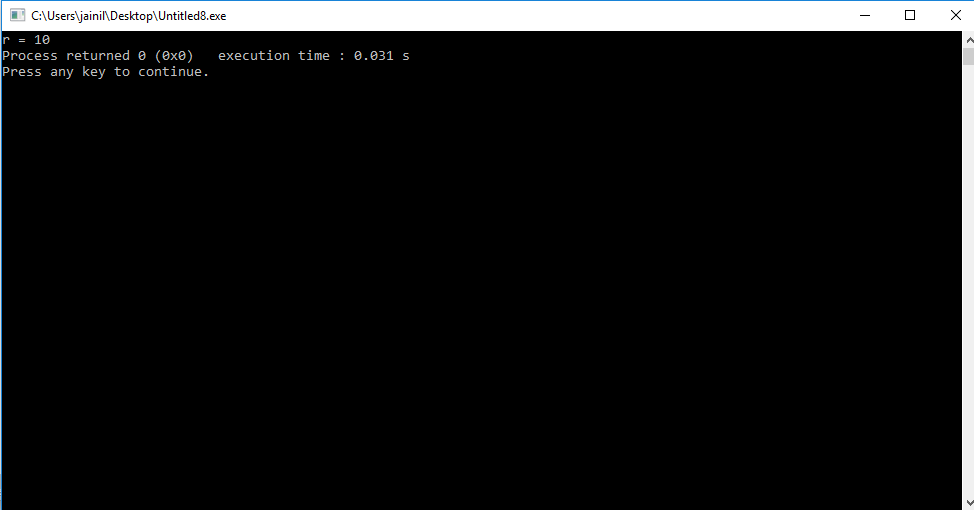
/\*int &c = NULL;

int &d[2] = {a,b};\*/

cout<<"r = "<< r;

return 0;

}



1>We cant deaclare reference without initialized here r is not initialized.

2>We cant declare null value to any reference.

3>we cant use reference as array.

**5. Find output of the following code: Explain how scope Resolution operator is used to access global version of a variable.**

#include<iostream.h>

#include<conio.h>

int m=30;

int main()

{

int m=20;

{

int m=10;

cout<<”we are in inner block”<<endl;

cout<<”value of m=”<<m<<”\n”;

cout<<”value of ::m=”<<::m<<”\n”;

}

cout<<”we are in outer block”<<endl;

cout<<”value of m=”<<m<<”\n”;

cout<<”value of ::m=”<<::m<<”\n”;

getch();

return 0;

}

PROGRAM::::::::

#include<iostream>

#include<conio.h>

using namespace std;

int m=30;

int main()

{

int m=20;

{

int m=10;

cout<<"we are in inner block"<<endl;

cout<<"value of m="<<m<<"\n";

cout<<"value of ::m="<<::m<<"\n";

}

cout<<"we are in outer block"<<endl;

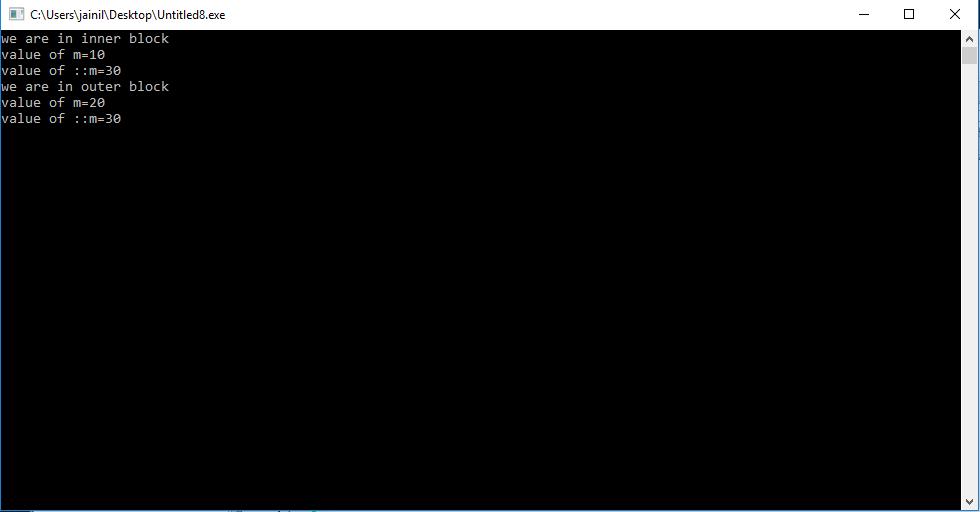
cout<<"value of m="<<m<<"\n";

cout<<"value of ::m="<<::m<<"\n";

getch();

return 0;

}



**6. Find Error in the following code of a program and give explanation why these errors exist.**

* //This is an example of **constant pointer**

#include <iostream>

using namespace std;

int main()

{

int var1 = 35,var2 = 20;

int \*const ptr = &var1;

ptr = &var2;

cout<<"var1= "<<\*ptr;

return 0;

}

PROGRAM::::::

#include <iostream>

using namespace std;

int main()

{

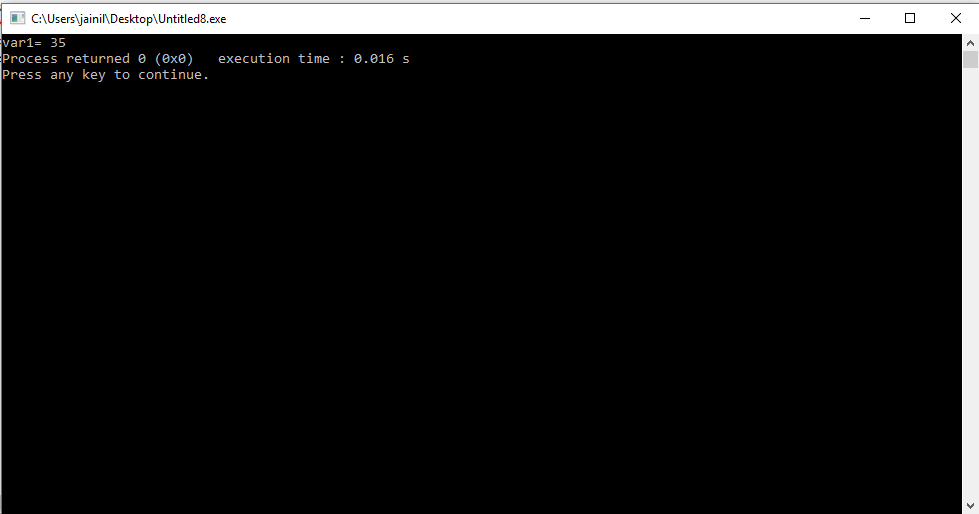
int var1 = 35,var2 = 20;

int \*const ptr = &var1;

cout<<"var1= "<<\*ptr;

return 0;

}



* //This is an example of **pointer to constant**

#include <iostream>

using namespace std;

int main()

{

int var1 = 43;

const int\* ptr = &var1;

\*ptr = 1;

var1=34;

cout<<"var1 = "<< \*ptr;

return 0;

}

Put comment infront of \*ptr = 1

Explain why value of var1 changes to 34 inspite ptr being pointer to constant ??

PROGRAM::::::::

//This is an example of pointer to constant

#include <iostream>

using namespace std;

int main()

{

int var1 = 43;

const int\* ptr = &var1;

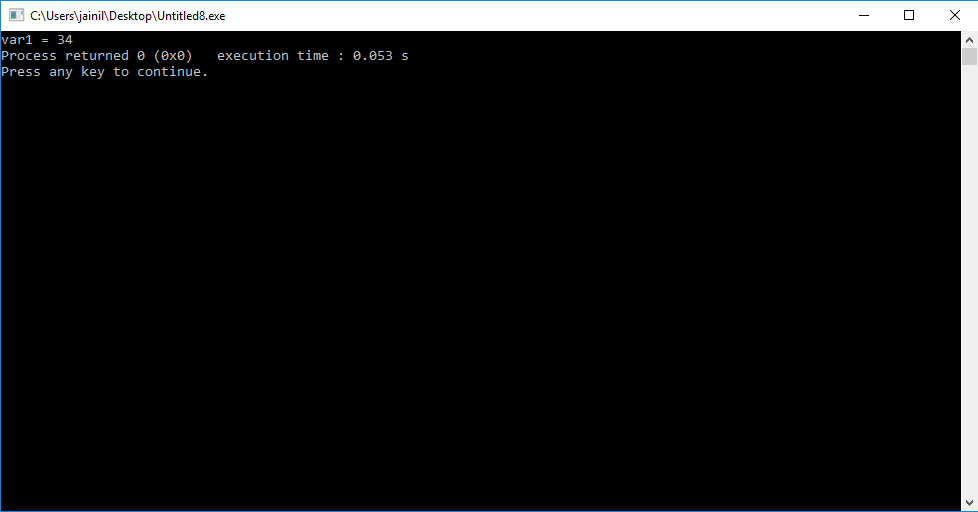
/\* \*ptr = 1; \*/

var1=34;

cout<<"var1 = "<< \*ptr;

return 0;

}



* //This is an example of **constant pointer to a constant**

#include <iostream>

using namespace std;

int main()

{

int var1 = 0,var2 = 0;

const int\* const ptr = &var1;

\*ptr = 1;

ptr = &var2;

cout<<"Var1 = "<<\*ptr;

return 0;

}

PROGRAM::::::

//This is an example of constant pointer to a constant

#include <iostream>

using namespace std;

int main()

{

int var1 = 0,var2 = 0;

const int\* const ptr = &var1;

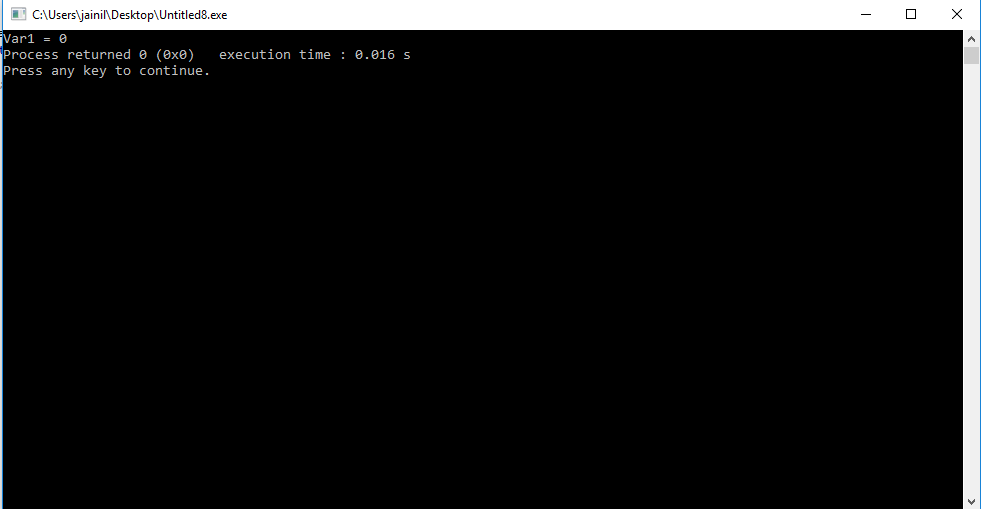
/\* \*ptr = 1;\*/

/\* ptr = &var2;\*/

cout<<"Var1 = "<<\*ptr;

return 0;

}



**7. Write a program to enter a size of array. Create an array of size given by user using “new” Dynamic memory management operator (free store operator). Enter the data to store in array and display the data after adding 2 to each element in the array. Delete the array by using “delete” memory management operator.**

#include<iostream>

using namespace std;

int main()

{

int n; int i=0;

cout<<"ENTER THE NUMBER OF ARRAY"<<endl;

int \*p,\*q;

cin>>n;

p=new int[n];

q=p;

for(i=0;i<n;i++)

{

cout<<"enter value of a\["<<i<<"\]"<<endl;

cin>>\*p;

p++;

}

p=q;

for(i=0;i<n;i++)

{

\*p=\*p+2;

p++;

}

cout<<"ADDING 2 TO ALL POINTER VARIABLE"<<endl;

p=q;

for(i=0;i<n;i++)

{

cout<<\*p<<endl;

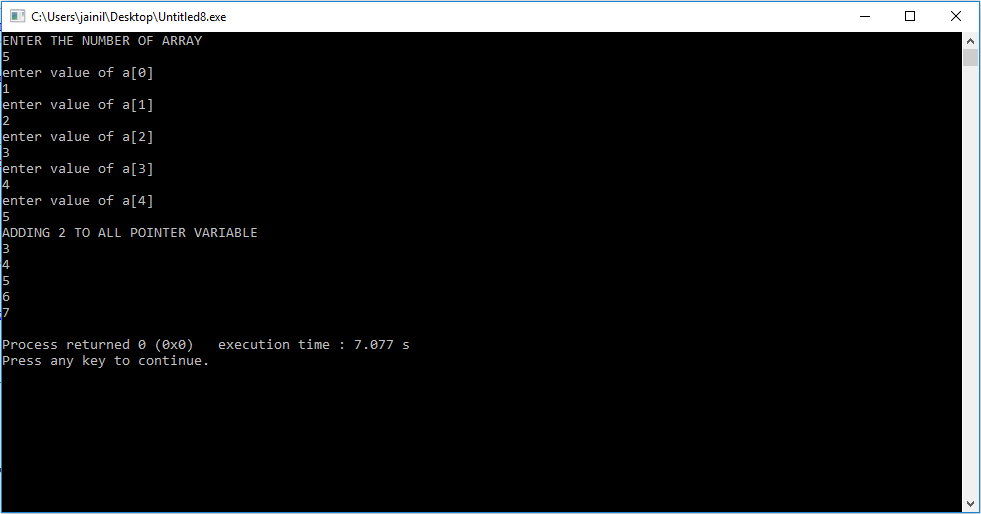
p++;

}

delete []p;

delete q;

}



**8. Find the output of following program. Explain the use of bool data type.**

#include<iostream>

using namespace std;

int main()

{

bool a = 321, b;

cout << "Bool a Contains : " << a<<endl;

int c = true;

int d = false;

cout<<"c = "<<c <<endl<<"d = "<<d;

c = a + a;

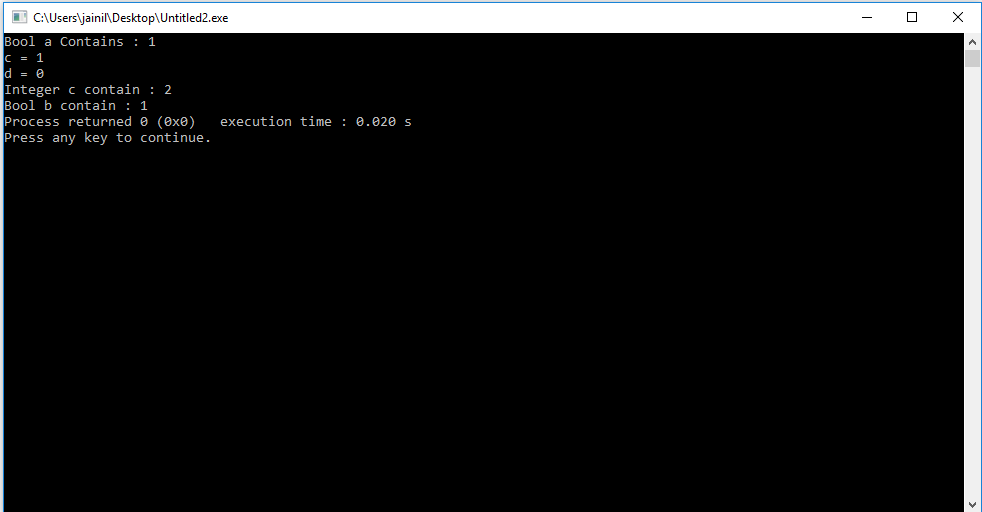
cout << "\nInteger c contain : " << c;

b = c + a;

cout << "\nBool b contain : " <<b;

return 0;

}



**9. Define three functions named convert (). First function takes rupees as an input argument and converts into paisa, second function takes meter as an input argument and converts into centimeter and last function takes character as an input argument and converts entered small character into capital. Here, rupee is in integer and meter is in float. Use concept of Function Overloading. Function overloading is also known as Compile Time Polymorphism or static binding.**

#include<iostream>

#include<cstdio>

#include<cctype>

using namespace std;

int convert(int a);

float convert(float b);

char convert(char c);

int main()

{

cout<<"enter rupees"<<endl;

int r,p;

cin>>r;

p=convert(r);

cout<<"paisa ="<<p<<endl;

cout<<"enter metre"<<endl;

int m,cm;

cin>>m;

cm=convert(m);

cout<<"centimetre="<<cm<<endl;

cout<<"enter char in lower case"<<endl;

char c,d;

cin>>c;

d=convert(c);

cout<<"upper case= "<<d<<endl;

}

int convert(int a)

{

return a\*100;

}

float convert(float b)

{

return b\*100;

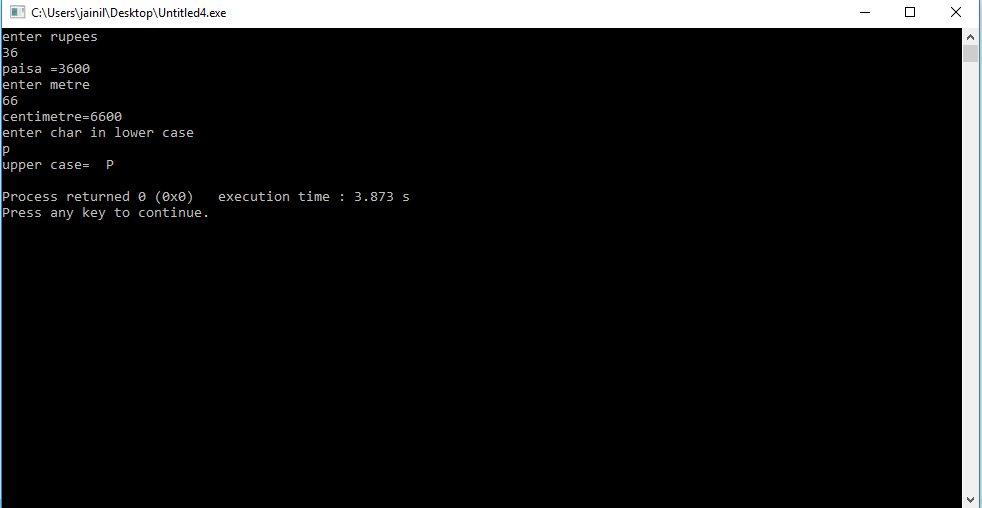
}

char convert(char c)

{

return c-32;

}



**10. Define four function void swap () which accepts two arguments by reference and swap the values. First function swaps two characters, second function swaps two integers, third function swaps two floats values and fourth function swaps two double values. Use the concept of call by reference in all four functions and function overloading.**

#include<iostream>

using namespace std;

void swap(char &a,char &b);

void swap(int &a,int &b);

void swap(float &a,float &b);

void swap(double &a,double &b);

int main()

{

int a,b;

cout<<"enter two integer"<<endl;

cin>>a>>b;

swap(a,b);

cout<<"swapped integer are"<<a<<" "<<b<<endl;

float e,f;

cout<<"enter two floating value"<<endl;

cin>>e>>f;

swap(e,f);

cout<<"swapped float are"<<e<<" "<<f<<endl;

double p,q;

cout<<"enter two double value"<<endl;

cin>>p>>q;

swap(p,q);

cout<<"swapped integer are"<<p<<" "<<q<<endl;

char x,y;

cout<<"enter two characters"<<endl;

cin>>x>>y;

swap(x,y);

cout<<"swapped integer are"<<x<<" "<<y<<endl;

return 0;

}

void swap(char &a,char &b)

{

char t;

t=a;

a=b;

b=t;

}

void swap(int &a,int &b)

{

int t;

t=a;

a=b;

b=t;

}

void swap(float &a,float &b)

{

float t;

t=a;

a=b;

b=t;

}

void swap(double &a,double &b)

{

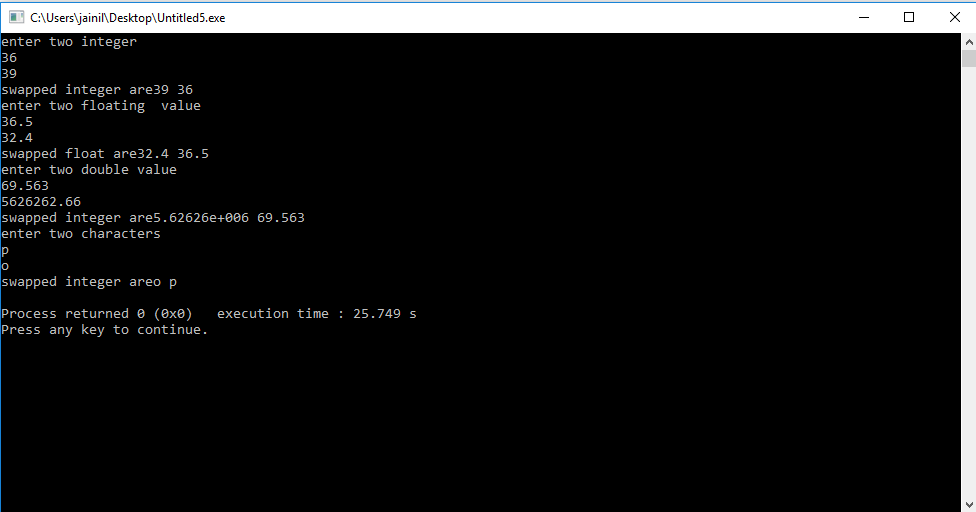
double t;

t=a;

a=b;

b=t;

}



**11. Declare an array of 5 double elements and assign some values to it. Create a function setvalue( ) which takes the index of the array as an input and Returns by Reference double value and change the value of the array element stored in that index.**

**For eg: double arr[] = {23.71, 45.6, 88.1, 19.1, 54.7}.**

**A function call such as setvalue (2) = 70.23 will result in the following array**

**double arr[] = {23.71, 45.6, 70.23, 19.1, 54.7} . (Hint: Declare array as a global variable).**

PROGRAM:::::

#include<iostream>

using namespace std;

double a[5];

double & setvalue(int i);

int main()

{

int num;

cout<<"enter a array of 5 double variable"<<endl;

cin>>a[0]>>a[1]>>a[2]>>a[3]>>a[4];

cout<<"\n\n\n"<<endl;

cout<<"enter a index to be changed"<<endl;

cin>>num;

setvalue(num);

cout<<"new array is:"<<endl;

int i;

for (i=0;i<5;i++)

{

cout<<a[i]<<" ";

}

}

double & setvalue(int i)

{

double t;

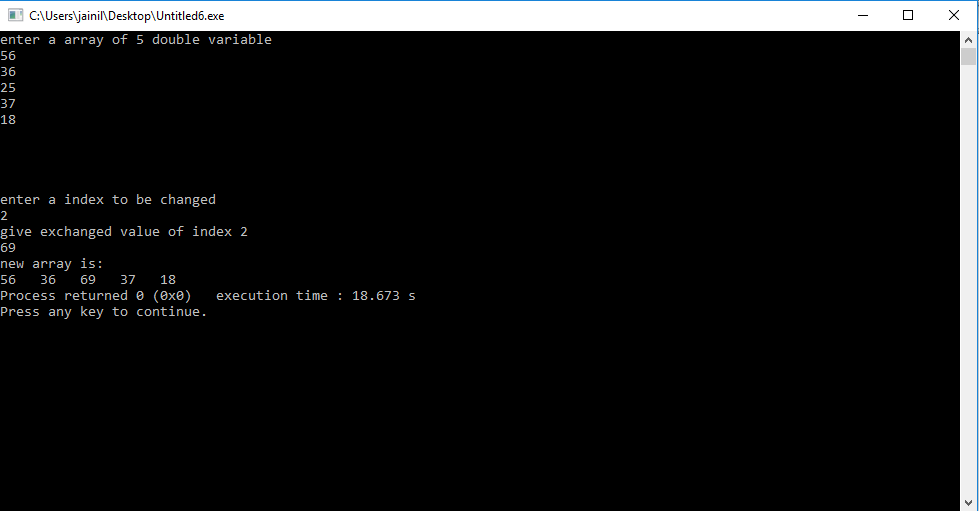
cout<<"give exchanged value of index "<<i<<endl;

cin>>t;

a[i]=t;

return a[i];

}



**12. Define three inline functions add (), subtract (), mul () which returns the addition, subtraction and multiplication of two numbers respectively.**

#include<iostream>

using namespace std;

inline int add(int,int);

inline int sub(int,int);

inline int mul(int,int);

int main()

{

int a,b,c;

cout<<"enter two number"<<endl;

cin>>a>>b;

c=add(a,b);

cout<<"addition is "<<c<<endl;

c=sub(a,b);

cout<<"subtraction is "<<c<<endl;

c=mul(a,b);

cout<<"multiplication is "<<c<<endl;

}

inline int add(int a,int b)

{

return a+b;

}

inline int sub(int a,int b)

{

return a-b;

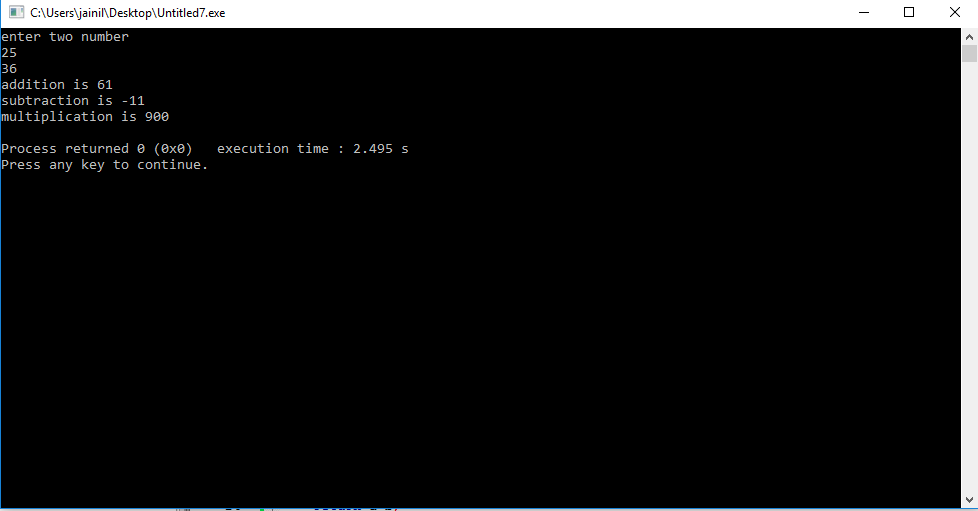
}

inline int mul(int a,int b)

{

return a\*b;

}



**13. Volume of Ellipsoid = (4/3) × pi × radius1 × radius2 × radius3. Write a**

**program having function volume () which takes three float arguments:**

**radius1, radius 2 and radius3 and returns the volume of an Ellipsoid.**

**Use default argument of 2 for radius1, 3 for radius2 and 4 for radius3**

**so that if arguments are omitted then the volume of Ellipsoid is always**

**100.48. Write a main ( ) function that gets values from the user to test**

**this function.**

PROGRAM:

#include<iostream>

#include<cstdio>

using namespace std;

inline float volume (float r1=2,float r2=3,float r3=4);

int main()

{

float r1,r2,r3,ans;

cout<<"enter radius 1"<<endl;

cin>>r1;

cout<<"enter radius 2"<<endl;

cin>>r2;

cout<<"enter radius 3"<<endl;

cin>>r3;

ans=volume(r1,r2,r3);

cout<<"volume is "<<ans;

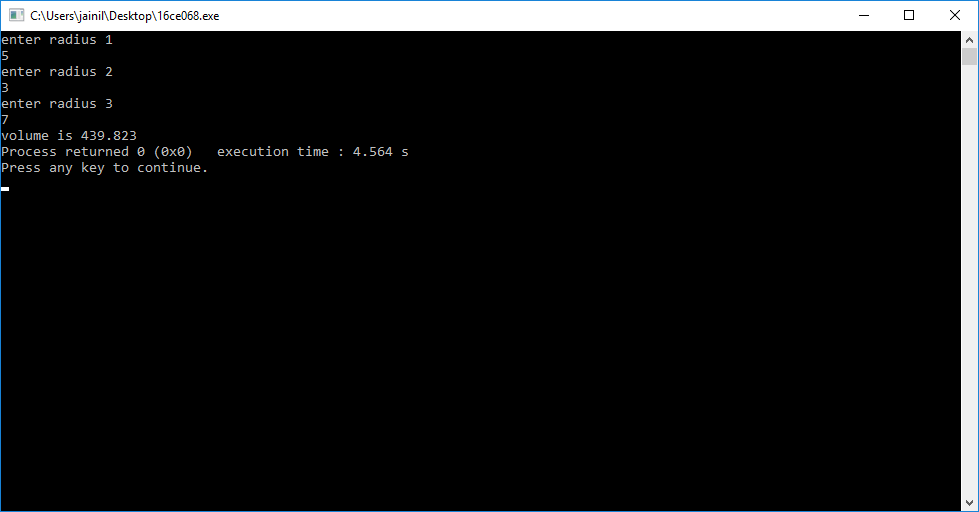
}

inline float volume (float r1,float r2,float r3)

{

return (((float)4/3)\*3.14159\*r1\*r2\*r3);

}



**14.** **Create a C++ Structure Weight having private data members: float**

**kg, grams. A member function getvalue () should enter their values.**

**Another member function putvalue () should display their values.**

#include<iostream>

using namespace std;

struct weight

{

private:

float kg;

float gm;

public:

void getvalue ()

{

cout<<"enter your weight in kg"<<endl;

cin>>kg;

cout<<"enter your weight in gram"<<endl;

cin>>gm;

}

void putvalue ()

{

cout<<endl<<"you have entered "<<kg<<" kg"<<endl;

cout<<endl<<"you have entered "<<gm<<" grams"<<endl;

}

};

int main()

{

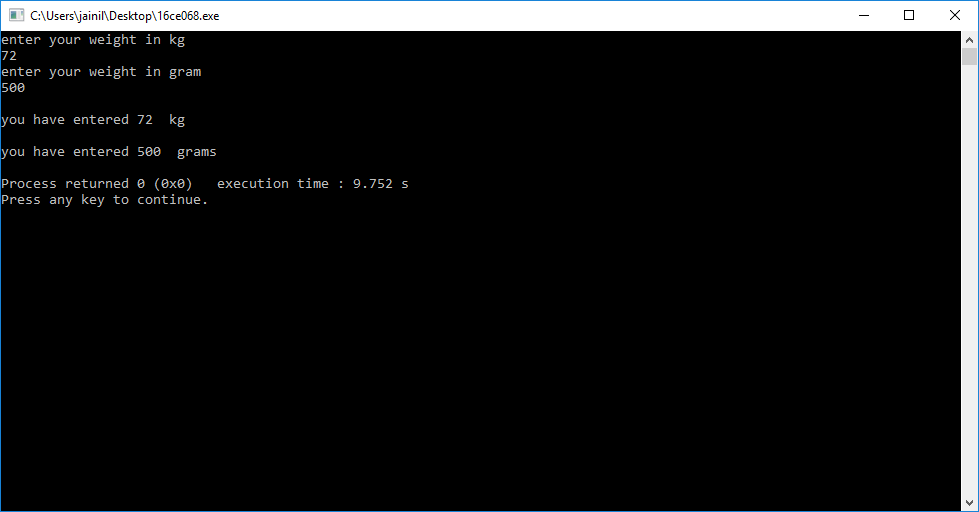
struct weight w;

w.getvalue();

w.putvalue();

return 0;

}



**15.** **Create a C++ Class Weight having private data members: float kg,**

**grams. A member function getvalue () should enter their values.**

**Another member function putvalue () should display their values.**

**Define both functions inside the class. Member function defined**

**inside the class behaves like an inline function. Illustrate the**

**difference between C++ Structure and C++ Class.**

#include<iostream>

using namespace std;

class weight

{

float kg;

float gm;

public:

void getvalue ()

{

cout<<"enter your weight in kg"<<endl;

cin>>kg;

cout<<"enter your weight in gram"<<endl;

cin>>gm;

}

void putvalue ()

{

cout<<endl<<"you have entered "<<kg<<" kg"<<endl;

cout<<endl<<"you have entered "<<gm<<" grams"<<endl;

}

};

int main()

{

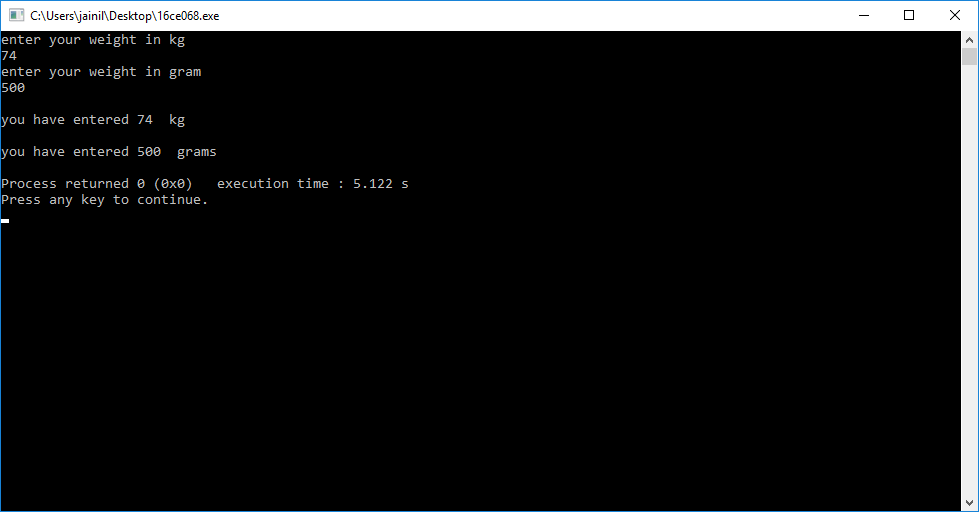
struct weight w;

w.getvalue();

w.putvalue();

return 0;

}



**16.** **Write a C++ program having class Batsman. It has private data**

**members: batsman name, bcode (4 Digit Code Number), innings, not**

**out, runs, batting average. Innings, not out and runs are in integer and**

**batting average is in float.**

**Define following function outside the class using scope resolution**

**operator.**

**1) Public member function getdata()to read values of data**

**members.**

**2) Public member function putdata()to display values of data]**

**members.**

**3) Private member function calcavg() which calculates the**

**batting average of a batsman. Also make this outside function**

**inline.**

**Hint : batting average = runs/(innings - notout)**

#include<iostream>

#include<cstdio>

using namespace std;

class batsman

{

char name[30];

int bcode,innings,notout,runs;

float battingaverage;

inline float calcavg(int runs,int innings,int notout);

public:

void getdata();

void putdata();

};

void batsman::getdata()

{

cout<<"enter name of batsman:";

cin>>name;

cout<<endl<<"enter bcode";

cin>>bcode;

cout<<endl<<"enter innings played";

cin>>innings;

cout<<endl<<"enter number of times not out";

cin>>notout;

cout<<endl<<"enter runs made by batsman";

cin>>runs;

}

void batsman::putdata()

{

cout<<"name of batsman:"<<name<<endl;

cout<<"bcode"<<bcode<<endl;

cout<<"innings played"<<innings<<endl;

cout<<"times not out"<<notout<<endl;

cout<<"total runs"<<runs<<endl;

cout<<"average is"<<calcavg(runs,innings,notout)<<endl;

}

inline float batsman::calcavg(int runs,int innings,int notout)

{

return ((float)runs/(innings-notout));

}

int main()

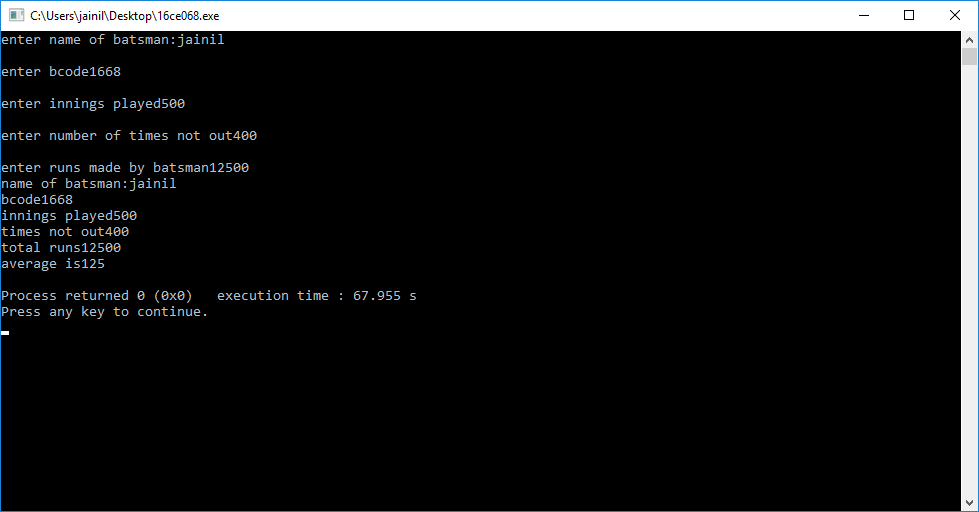
{

batsman b;

b.getdata();

b.putdata();

}



**17.** **Write a C++ program having class Dist with private data member int**

**feet and float inches. Define following public member functions for it.**

**1) getdata( ) to take feet and inches as input2) putdata( ) to display distance in 1’2.5” format**

**3) add( )to do addition of two distances such that it can handle**

**function call**

**d1.add(d2) where d1, d2 and d3 are objects of class.**

**Use Concept of Object as Function Arguments.**

#include<iostream>

#include<cstdio>

using namespace std;

class dist

{

float inches;

int feet;

public:

void getdata()

{

cout<<"enter feet"<<endl;

cin>>feet;

cout<<"enter inches"<<endl;

cin>>inches;

}

void putdata()

{

cout<<feet<<"'"<<inches<<"''";

}

void add(dist a);

};

void dist::add(dist a)

{

dist c;

c.feet=feet+a.feet;

c.inches=inches+a.inches;

c.putdata();

}

int main()

{

dist d1,d2,d3;

d1.getdata();

d2.getdata();

d1.putdata();

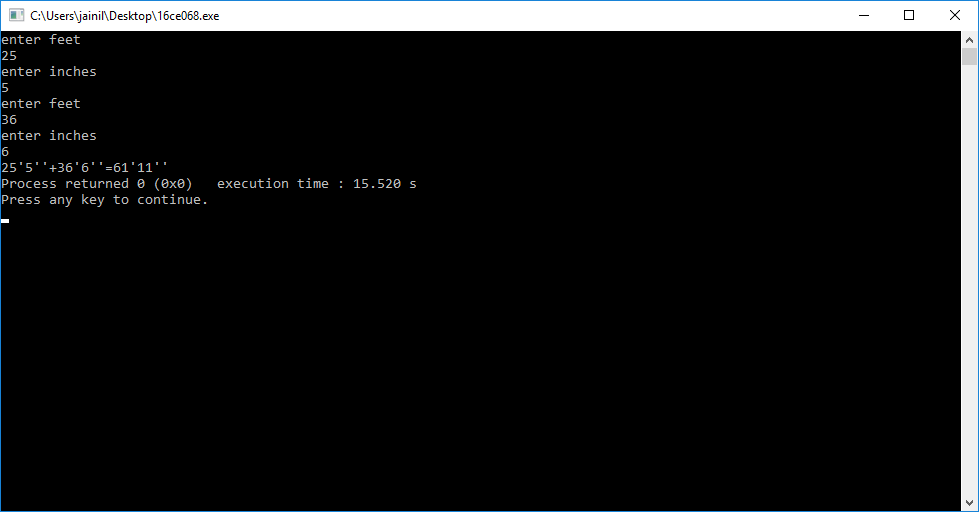
cout<<"+";

d2.putdata();

cout<<"=";

d1.add(d2);

}



**18.** **In above program apply function overloading in an Class by**

**implementing two add() function to do addition of two distance which**

**can support following function calls.**

**(i) d3 . add ( d1 , d2 )**

**(ii) d3 = d1 . add( d2 )**

**Use concepts of Object as Function Arguments and function returning object.**

#include<iostream>

#include<cstdio>

using namespace std;

class dist

{

float inches;

int feet;

public:

void getdata()

{

cout<<"enter feet"<<endl;

cin>>feet;

cout<<"enter inches"<<endl;

cin>>inches;

}

void putdata()

{

cout<<feet<<"'"<<inches<<"''";

}

dist add(dist a);

void add(dist a,dist b);

};

dist dist::add(dist a)

{

dist c;

c.feet=feet+a.feet;

c.inches= inches+a.inches;

return c;

}

void dist::add(dist a,dist b)

{

inches=a.inches+b.inches;

feet=a.feet+b.feet;

putdata();

}

int main()

{

dist d1,d2,d3,d4;

d1.getdata();

d2.getdata();

d1.putdata();

cout<<"+";

d2.putdata();

cout<<"=";

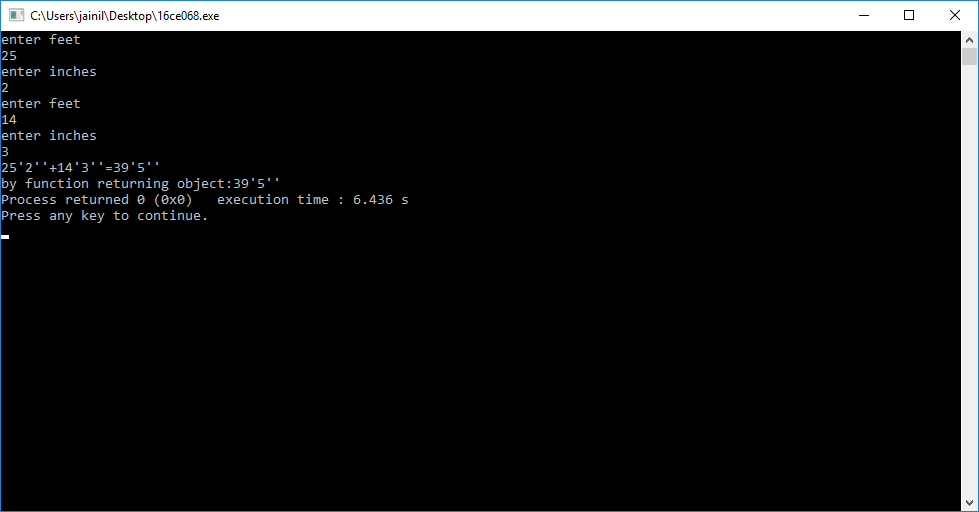
d3.add(d1,d2);

cout<<endl<<"by function returning object:";

d4=d1.add(d2);

d4.putdata();

}



**19.** **Define a class Dist with int feet and float inches. Define member**

**function that displays distance in 1’-2.5” format. Also define member**

**function scale ( ) function that takes object by reference and scale**

**factor in float as an input argument. The function will scale the**

**distance accordingly.**

**For example, 20’-5.5” and Scale Factor is 0.5 then answer is 10’-2.75”.**

#include<iostream>

#include<cstdio>

using namespace std;

class dist

{

float inches;

float feet;

public:

void getdata()

{

cout<<"enter feet"<<endl;

cin>>feet;

cout<<"enter inches"<<endl;

cin>>inches;

}

void putdata()

{

cout<<feet<<"'"<<inches<<"''";

}

void scale(dist &a,float b)

{

a.feet\*=b;

a.inches\*=b;

}

};

int main()

{

dist d1;

float f;

d1.getdata();

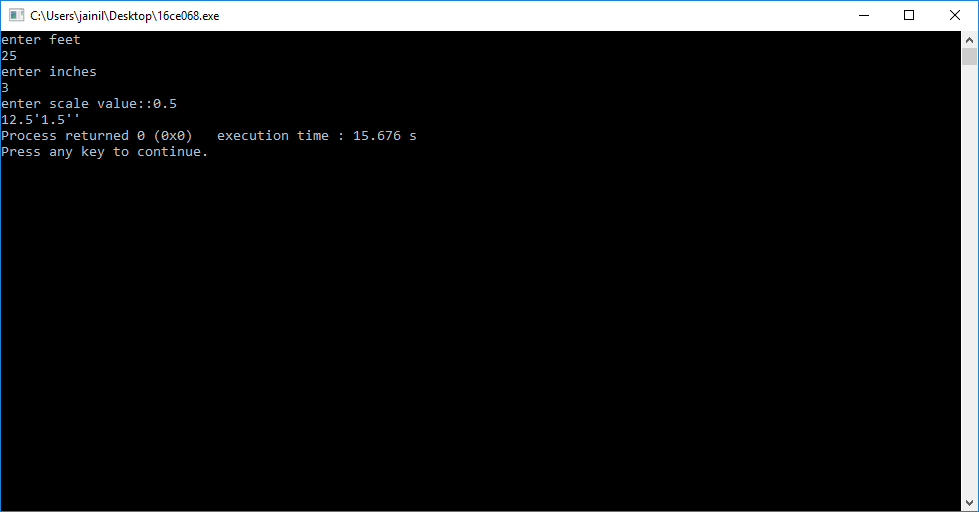
cout<<"enter scale value::";

cin>>f;

d1.scale(d1,f);

d1.putdata();

}



**20.Define class *Mobile* having data members: year of manufacture, price. A class has member functions to get and print data. Define member function validity ( ) that checks whether the mobile is older than 2005 or not. If mobile is older, it is invalid. Input data for 4 mobiles in main ( ) and display information about the phone which is invalid. Use the concept of Array of Objects**

#include<iostream>

#include<cstdio>

using namespace std;

class mobile

{

int year;

float price;

public:

void getdata()

{

cout<<"enter year of manufacturing:";

cin>>year;

cout<<"enter price:";

cin>>price;

}

void putdata()

{

cout<<"year:"<<year<<" "<<"price:"<<price<<endl;

}

int validity()

{

if(year<2005)

{

return 1;

}

else

{

return 0;

}

}

};

int main()

{

mobile m[4];

for(int i=0;i<4;i++)

{

m[i].getdata();

}

cout<<"OLDER MOBILES ARE :"<<endl;

for(int i=0;i<4;i++)

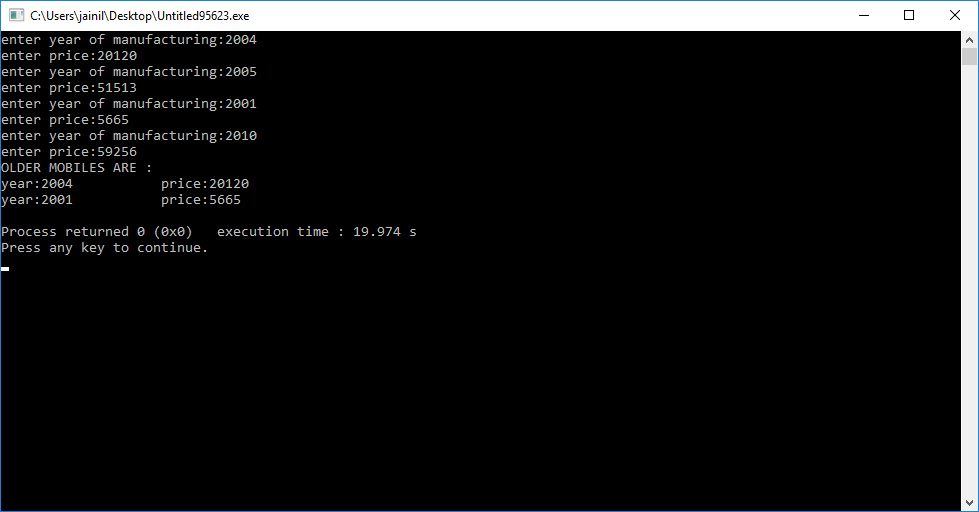
{

if((m[i].validity())==1)

{

m[i].putdata();

}

}} 

**21. Create a Class Gate for students appearing in Gate (Graduate Aptitude test for Engineering) exam. There are three examination center Vadodara, Surat, and Ahmedabad where Gate exams are conducted. A class has data members: Registration number, Name of student, Examination center. Class also Contains static data member ECV\_Cnt, ECS\_Cnt and ECA\_Cnt which counts the number of students in Vadodara, Surat and Ahmedabad exam center respectively. Class Contains two Member function getdata () which gets all information of students and counts total students in each exam center and pudata () which prints all information about the students. Class also contains one static member function getcount () which displays the total number of students in each examination center.  Write a program for 5 students and display the total number of students in each examination center. Use static data member, static member function and Array of Objects.**

#include<iostream>

#include<cstring>

using namespace std;

class gate

{

static int ecv\_cnt;

static int ecs\_cnt;

static int eca\_cnt;

long int number;

char name[30];

string area;

int centre;

public:

void getdata()

{

cout<<"enter registration number:";

cin>>number;

cout<<"enter name:";

cin>>name;

start:

cout<<"enter your exam centre:";

cin>>area;

if(area!="vadodara"&&area!="ahmedabad"&&area!="surat")

{

cout<<"please enter correct exam centre \n you can select from\n1.ahmedabad\n2.surat\n3.vadodara"<<endl;

goto start;

}

if(area=="vadodara")

ecv\_cnt+=1;

if(area=="surat")

ecs\_cnt+=1;

if(area=="ahmedabad")

eca\_cnt+=1;

}

void getcount()

{

cout<<"total number of students in vadodara is:"<<ecv\_cnt<<endl;

cout<<"total number of students in surat is:"<<ecs\_cnt<<endl;

cout<<"total number of students in ahmedabad is:"<<eca\_cnt<<endl;

}

};

int gate::ecv\_cnt;

int gate::ecs\_cnt;

int gate::eca\_cnt;

int main()

{

int i;

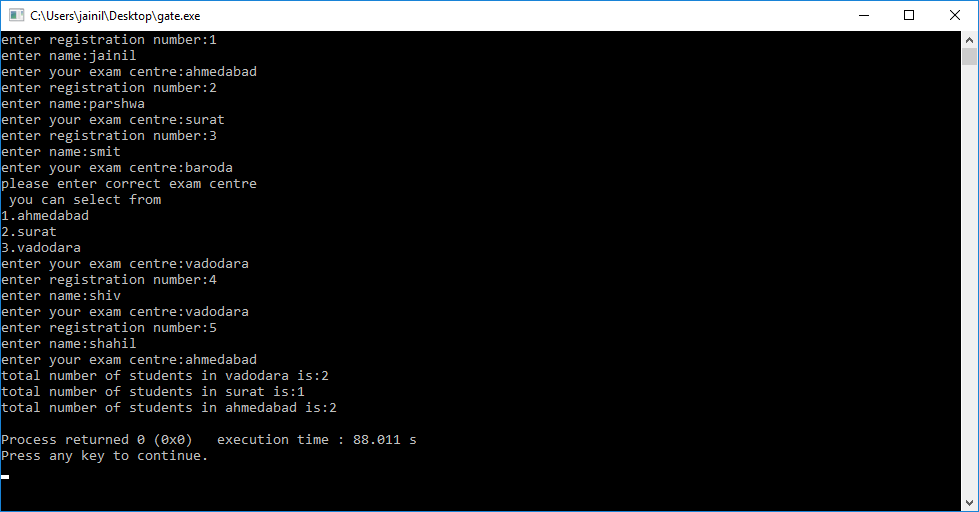
gate g[5];

for(i=0;i<5;i++)

{ g[i].getdata();}

g[0].getcount();

}



**22.Create a class Corporate which contains name of employee and salary in float. Create another class Education which contains name of employee and salary in float. Both the class have member function to get and print data. Write a non-member function compare() which receives object of both the classes and compares the salary of employee in education and corporate field. Use the concept of friend function.**

#include<iostream>

using namespace std;

class education;

class corporate

{

string nameofemployee;

float salary;

public:

friend compare(corporate c,education e);

getdata()

{

cout<<"enter name of corporate employee"<<endl;

cin>>nameofemployee;

cout<<"enter salary of corporate employee"<<endl;

cin>>salary;

}

putdata()

{

cout<<"name:"<<nameofemployee<<" salary:"<<salary<<endl;

}

};

class education

{

string nameofemployee;

float salary;

public:

friend compare(corporate c,education e);

getdata()

{

cout<<"enter name of corporate employee"<<endl;

cin>>nameofemployee;

cout<<"enter salary of corporate employee"<<endl;

cin>>salary;

}

putdata()

{

cout<<"name:"<<nameofemployee<<" salary:"<<salary<<endl;

}

};

compare(corporate c,education e)

{

if(c.salary==e.salary)

{

cout<<"both salary are equal";

}

else if(c.salary>e.salary)

{

cout<<"corporate salary is higher than education salary"<<endl;

c.putdata();

}

else

{

cout<<"education salary is higher than corporate salary"<<endl;

e.putdata();

}

}

int main()

{

corporate c;

education e;

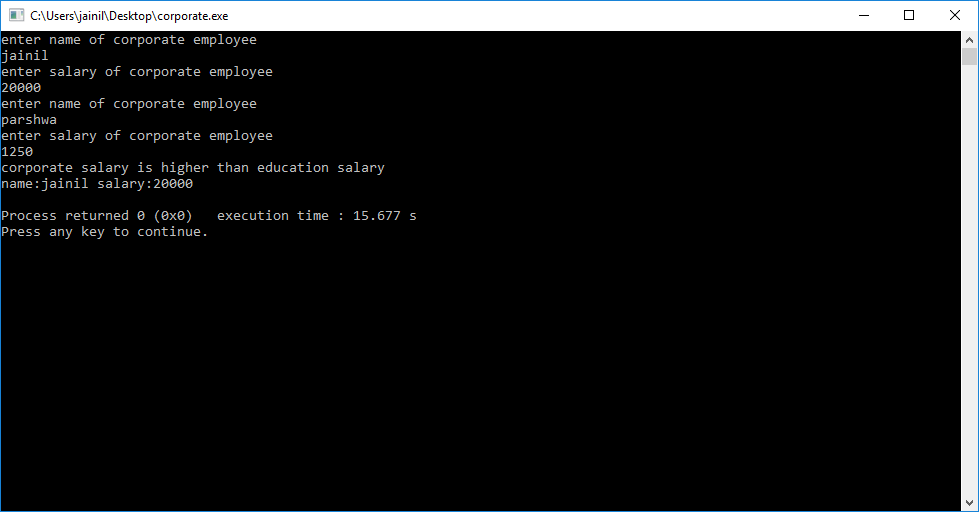
c.getdata();

e.getdata();

compare(c,e);

return 0;

}



**23.Create a Class Date having data members: int dd, mm, yyyy. Class has one member function to input the dates and another member function which prints the dates. Write a main() function which takes two dates as input. Write a friend function swapdates() which takes two objects by reference of type Date and swaps both the dates. Use the concept of Friend function which takes objects by reference.**

#include<iostream>

using namespace std;

class date

{ int dd,mm,yyyy;

public:

getdata()

{ cout<<"enter date in dd mm yyyy format:";

cin>>dd>>mm>>yyyy;}

putdata()

{ cout<<dd<<":"<<mm<<":"<<yyyy<<endl;}

friend swapdates(date &a,date &b)

{ date t;

t=a;

a=b;

b=t; }

};

int main()

{ date a,b;

a.getdata();

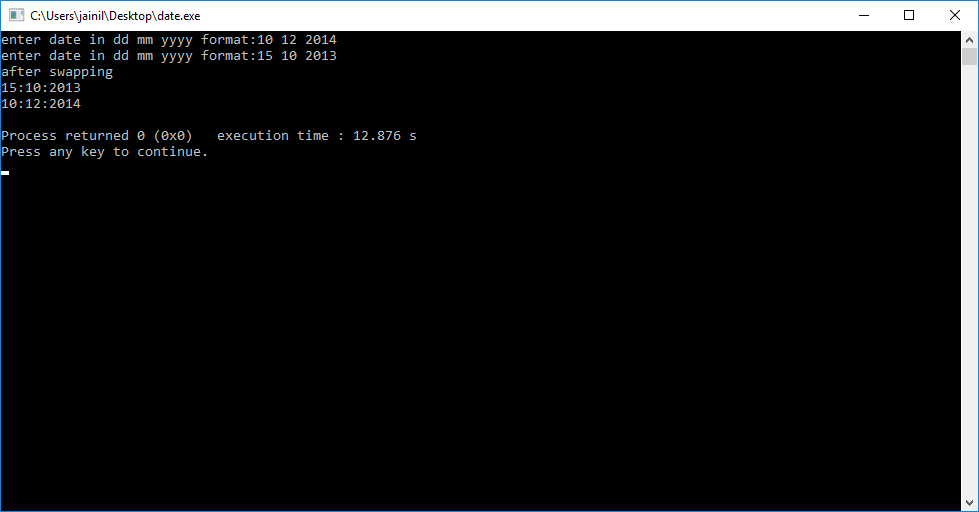
b.getdata();

swapdates(a,b);

cout<<"after swapping"<<endl;

a.putdata();

b.putdata();}



**24.Create a class Customer having data members: name of the customer and customer number in integer and member function to get customer data. Create another class Manager having data members: name of manager and employee id in integer and member function to get managers data. Class Manager also have member function get\_cust\_data () which takes objects of class Customer as input and prints the customers details and is a friend function of class Customer . Write a main () function to test all this function. Use the concepts of Member function of one class can be a Friend Function of another class.**

#include<iostream>

using namespace std;

class customer;

class manager

{

char name[30];

int id;

public:

void getdata()

{

cout<<"Enter manager name ";

cin>>name;

cout<<"Enter manager number ";

cin>>id;

}

void get\_cust\_data(customer a);

};

class customer

{

char name[30];

int cno;

public:

void getdata()

{

cout<<"Enter customer name ";

cin>>name;

cout<<"Enter customer number ";

cin>>cno;

}

friend void manager::get\_cust\_data(customer a);

};

void manager::get\_cust\_data(customer a)

{

cout<<"The customer name is "<<a.name<<endl;

cout<<"the customer number is "<<a.cno<<endl;

}

int main()

{

customer a,a2;

manager b,b2;

a.getdata();

b.getdata();

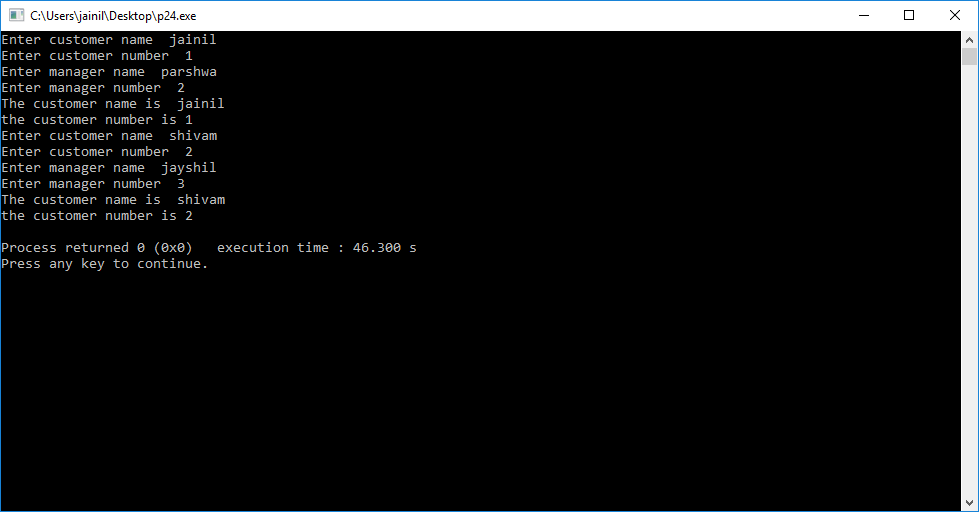
b.get\_cust\_data(a);

a2.getdata();

b2.getdata();

b2.get\_cust\_data(a2);

}



**25.Create a class Child having data members: name of the child and gender and a member function to get and print child data. Create another class Parent which is a friend class of child class. Class Parent have member function ReadChildData() which takes child’s object by reference as input argument and Reads the childs data and DisplayChildData() which takes childs object as argument and displays childs data. Use the concepts of Friend Class.**

#include<iostream>

#include<cstdio>

using namespace std;

class parent;

class child

{

string name;

string gender;

friend class parent;

void getdata()

{

cout<<"ENTER CHILD NAME:";

cin>>name;

cout<<endl;

cout<<"ENTER GENDER";

cin>>gender;

cout<<endl;

}

void putdata()

{

cout<<"CHILD NAME:"<<name;

cout<<endl;

cout<<"GENDER:"<<gender;

cout<<endl;

}

};

class parent

{

public:

void readchilddata(child &a)

{

a.getdata();

}

void displaychilddata(child &b)

{

b.putdata();

}

};

int main()

{

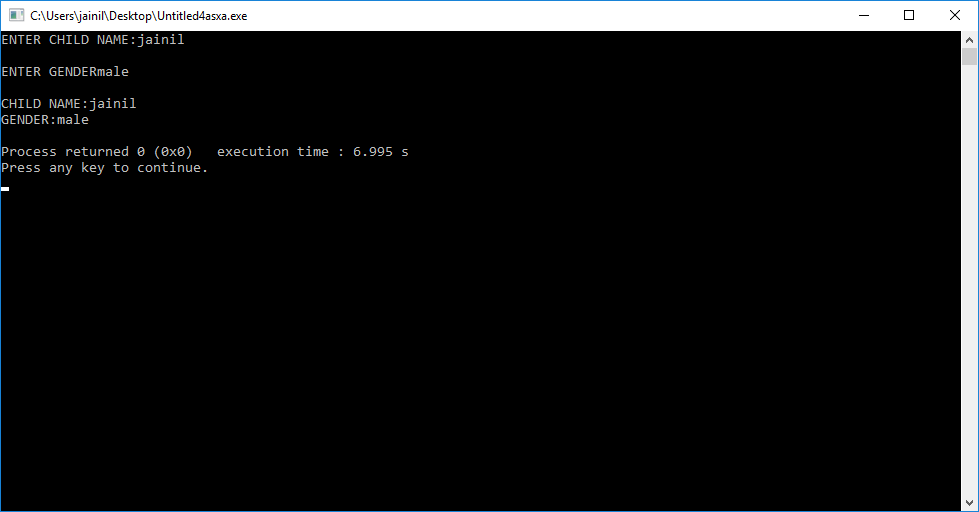
child c;

parent p;

p.readchilddata(c);

p.displaychilddata(c);

}



**26. Check the following C++ code and find if there is any error in code, give justification for the error, correct the code and write the output:**

**// This is an example of const member functions**

**#include<iostream>**

**using namespace std;**

**class sample**

**{**

**int m, n;**

**public:**

**void getdata();**

**void putdata() const;**

**};**

**void sample::getdata()**

**{**

**cout<< "Enter m & n";**

**cin>>m>>n;**

**}**

**void sample::putdata() const**

**{**

**m=12;**

**n=34;**

**cout<< " m = "<<m<<"n= "<<n;**

**}**

**int main()**

**{**

**sample s1;**

**s1.getdata();**

**s1.putdata();**

**return 0;**

**}**

// This is an example of const member functions

#include<iostream>

using namespace std;

class sample

{

int m, n;

public:

void getdata();

void putdata() const;

};

void sample::getdata()

{

cout<< "Enter m & n"<<endl;

cin>>m>>n;

}

void sample::putdata() const

{

cout<< " m = "<<m<<"n= "<<n;

}

int main()

{

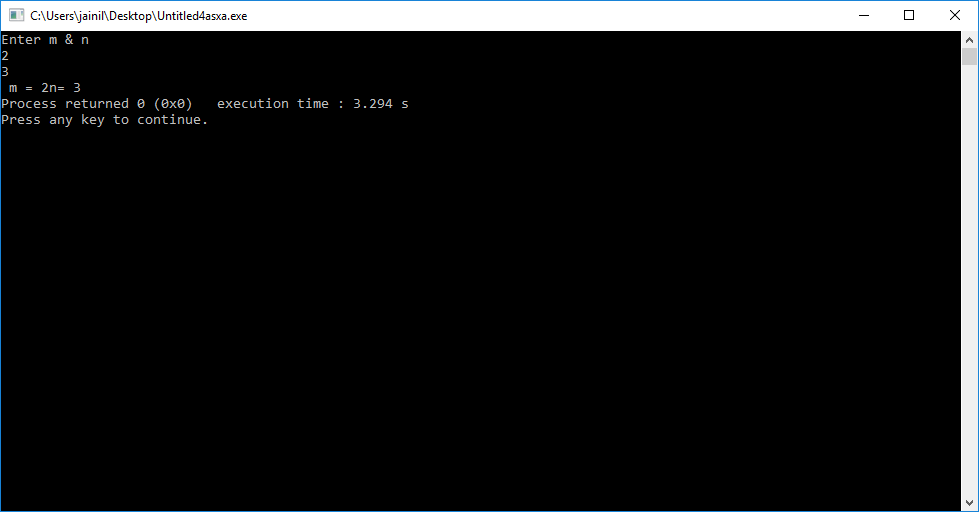
sample s1;

s1.getdata();

s1.putdata();

return 0;

}



**2.// This is an example of (a) Pointer to data members,**

**(b)Pointer to member functions**

**(A)#include<iostream>**

**using namespace std;**

**class student**

**{**

**public:**

**int roll\_no;**

**};**

**int main()**

**{**

**// declaring pointer to data member**

**int student :: \*p1 = &student::roll\_no;**

**student s;**

**student \*optr = &s;**

**s->\*p1 = 42;**

**cout<<"Roll no is "<<s->\*p1<<endl;**

**optr.\*p1 = 45;**

**cout<<"Roll no is "<<optr.\*p1<<endl;**

**return 0;}**

#include<iostream>

using namespace std;

class student

{

public:

int roll\_no;

};

int main()

{

// declaring pointer to data member

int student :: \*p1 = &student::roll\_no;

student s;

student \*optr = &s;

optr->\*p1 = 42;

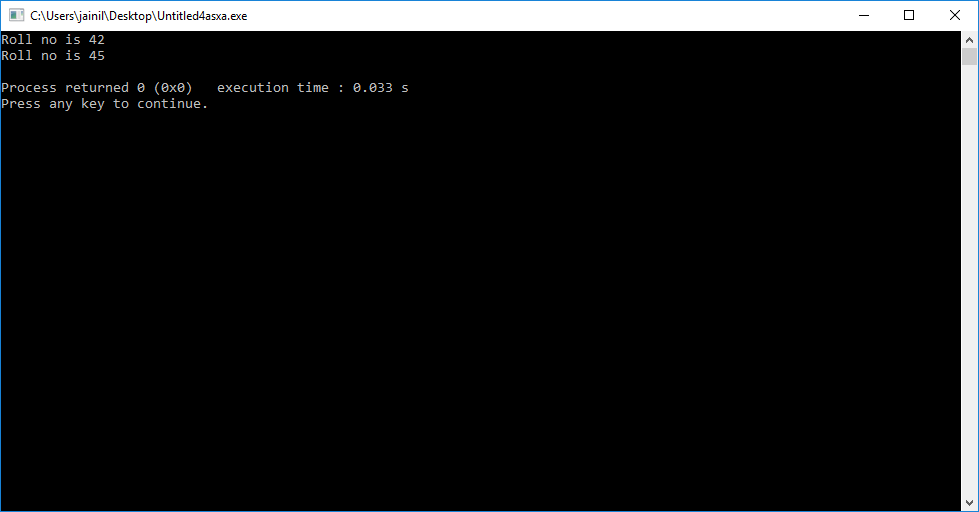
cout<<"Roll no is "<<s.\*p1<<endl;

optr->\*p1 = 45;

cout<<"Roll no is "<<optr->\*p1<<endl;

return 0;

}



**(b)#include<iostream>**

**class employee**

**{**

**public:**

**void hello()**

**{**

**cout<<"Hi hello"<<endl;**

**}**

**};**

**int main()**

**{**

**// declaring pointer to member function hello**

**void (employee ::\*fp)() = &employee::hello;**

**employee e;**

**employee \*optr = &e;**

**(e->\*fp)();**

**(optr.\*fp)();**

**return 0;**

**}**

#include<iostream>

using namespace std;

class employee

{

public:

void hello()

{

cout<<"Hi hello"<<endl;

}

};

int main()

{

// declaring pointer to member function hello

void (employee ::\*fp)() = &employee::hello;

employee e;

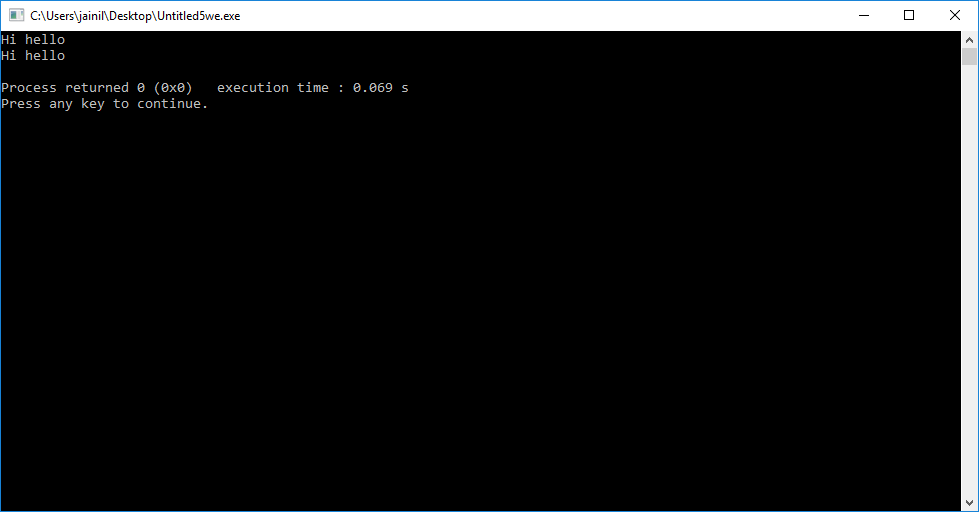
employee \*optr = &e;

(e.\*fp)();

(optr->\*fp)();

return 0;

}



**3.)// This is an example of Local Classes**

**#include<iostream>**

**using namespace std;**

**namespace std;**

**void testlocalclass()**

**{**

**class Test**

**{**

**static int cnt;**

**public:**

**void set()**

**{cout<<"Enter Count: "; cin>>cnt; }**

**void get();**

**};**

**void Test:: get()**

**{ cout<<"Count: = " <<cnt; }**

**Test t;**

**t.set();**

**t.get();**

**}**

**int main()**

**{**

**testlocalclass();**

**return 0;**

**}**

#include<iostream>

using namespace std;

void testlocalclass()

{

class Test

{

int cnt;

public:

void set()

{cout<<"Enter Count: "; cin>>cnt; }

void get()

{ cout<<"Count: = " <<cnt; }

};

Test t;

t.set();

t.get();

}

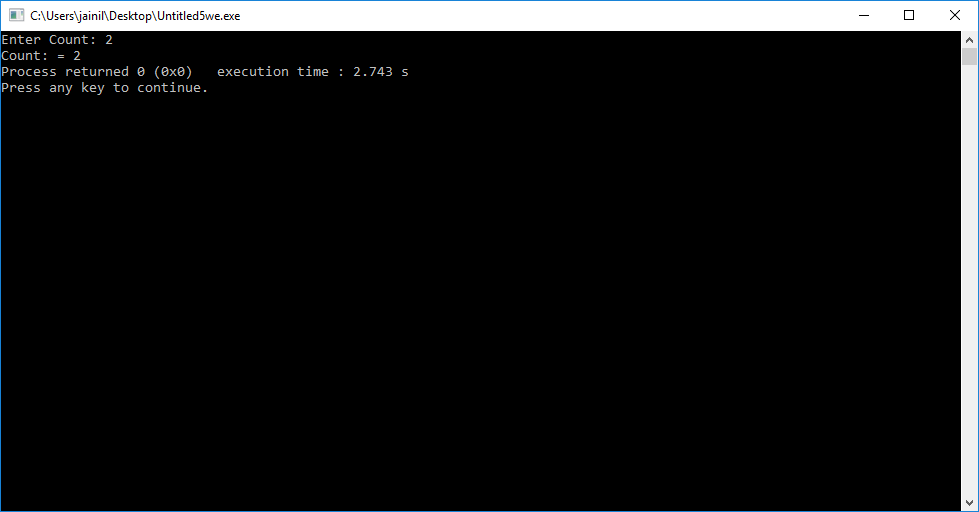
int main()

{

testlocalclass();

return 0;

}



**27.Write a C++ program having class time with data members: hr, min and sec. Define following member functions.**

1. **getdata() to enter hour, minute and second values**
2. **putdata() to print the time in the format 11:59:59**
3. **default constructor**
4. **parameterized constructor**
5. **copy constructor**
6. **Destructor.**

**Use 52 as default value for sec in parameterized constructor.**

**Use the concepts of default constructor, parameterized constructor, Copy constructor, constructor with default arguments and destructor.**

#include<iostream>

using namespace std;

class time

{

int hr,minute,sec;

public:

void getdata()

{

cout<<"enter hr min sec"<<endl;

cin>>hr>>minute>>sec;

}

void putdata()

{

cout<<hr<<":"<<minute<<":"<<sec<<endl;

}

time()

{hr=0;minute=0;sec=0;}

time(int a,int b,int c)

{

hr=a;

minute=b;

sec =c;

}

time(time &a)

{

hr=a.hr;

minute=a.minute;

sec=a.sec;

}

~time()

{

cout<<"object is destroyed"<<endl;

}

};

int main()

{

int a,b,c;

cout<<"default constructor"<<endl;

time t1;

t1.putdata();

cout<<"parameterised constuctor"<<endl;

cout<<"enter hr min sec"<<endl;

cin>>a>>b>>c;

time t2(a,b,c);

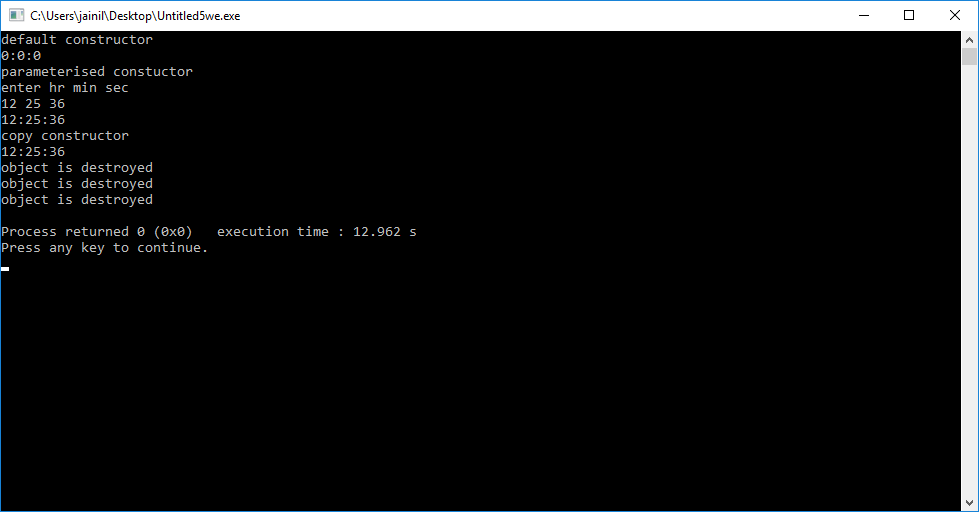
t2.putdata();

cout<<"copy constructor"<<endl;

time t3(t2);

t3.putdata();

}



**28. Create a class Number having int num as member. The class has input and output functions. Overload unary operator (++) such that it supports N1=N2++ and N3=++N1 and Overload unary (-) such that it supports**

**N3 = - N3. Also define default, parameterized and copy constructor for the class. Also explain use of nameless object in operator overloading. Use the concept of Overloading Unary Operators. Operator overloading is also known as Compile Time Polymorphism or static binding.**

#include<iostream>

using namespace std;

class number

{

public:

int num;

number()

{

num=0;

}

number(int n)

{

num=n;

}

void getdata()

{

cout<<"enter num"<<endl;

cin>>num;

}

void putdata(int i)

{

cout<<i<<"::"<<num<<endl;

}

number operator ++(int x)

{

x=num++;

return (number(x));

}

number operator ++()

{

int x;

x=++num;

return(number(x));

}

number operator -()

{

int x;

x=-num;

return (number(x));

}

};

int main()

{

number n1,n2,n3;

n2.getdata();

n1=n2++;

cout<<"after n1=n2++"<<endl;

n1.putdata(1);

n2.putdata(2);

n3=++n1;

cout<<"after n3=++n1"<<endl;

n1.putdata(1);

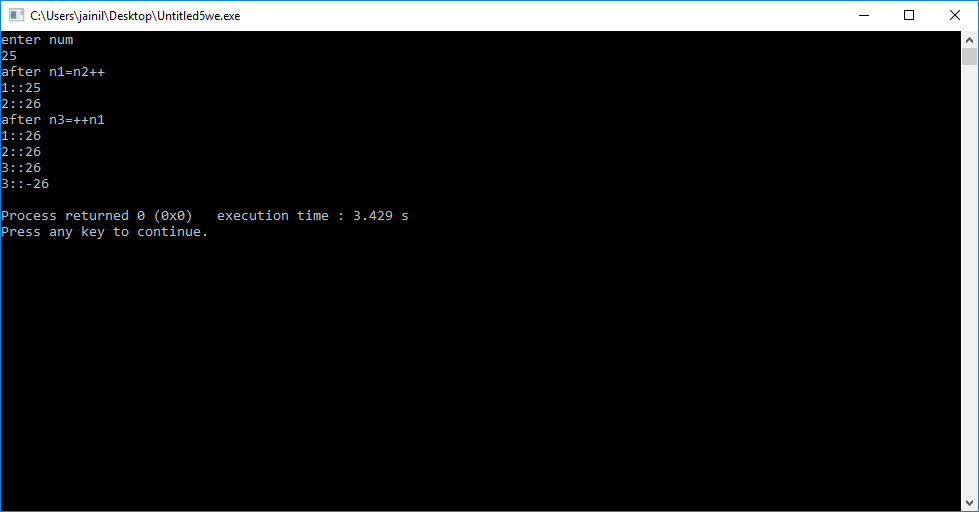
n2.putdata(2);

n3.putdata(3);

n3=-n3;

n3.putdata(3);

}



**29. Create a class complex having data members int real , img and member function to  print data. Overload Unary operator (-) using friend function such that it supports –C1 where C1 is the object of class complex. Also define default, parameterized and copy constructor for the class.Use the concept of Overloading Unary Operators with friend function.**

#include<iostream>

using namespace std;

class complex

{

int real,img;

public:

void getdata()

{ cout<<"enter real and imaginary part"<<endl;

cin>>real>>img;

}

void putdata()

{ cout<<real<<"+"<<"("<<img<<")"<<"i";}

complex(){real=0,img=0;}

complex(int x,int y){real=x;img=y;}

friend void operator -(complex &c);

};

void operator -(complex &c)

{ c.real=-c.real;

c.img=-c.img;}

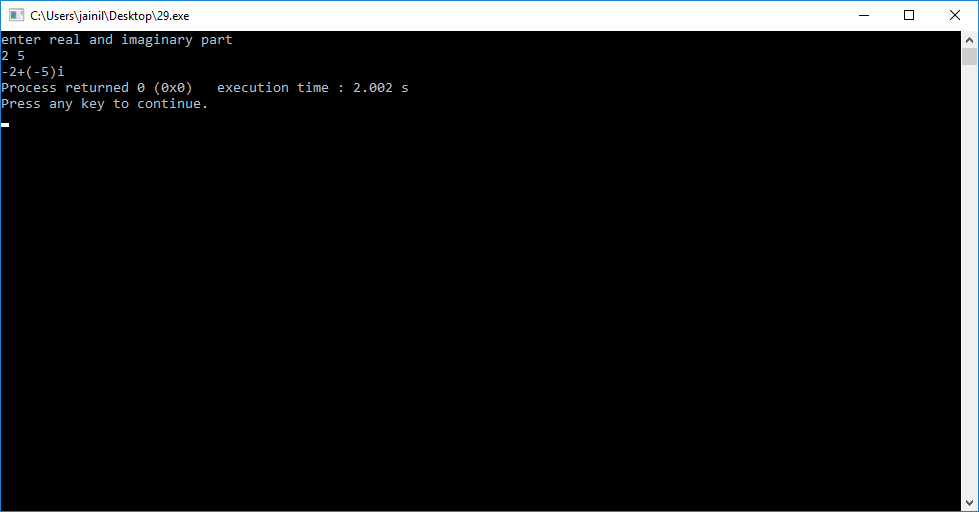
int main()

{ complex c1;

c1.getdata();

-c1;

c1.putdata();}



**30.Define a class NUM having two integer members a and b. A class has member functions for input and output. Overload the operators – and = = such that it supports the following  statements:**

**NUM N1, N2, N3; N3=N1-N2; if(N1==N2){…}**

**Use the concept of Overloading Binary Operators.**

#include<iostream>

#include<cstdio>

using namespace std;

class num

{

int a;

int b;

public:

void getdata()

{

cout<<"enter a and b"<<endl;

cin>>a>>b;

}

void putdata()

{

cout<<"a= "<<a<<" b= "<<b<<endl;

}

num operator -(num n)

{

num sub;

sub.a=a-n.a;

sub.b=b-n.b;

return sub;

}

int operator ==(num n2)

{

if(a==n2.a&&b==n2.b)

{

return 1;

}

else

{

return 0;

}

}

};

int main()

{

num n1,n2,n3;

cout<<"for number 1"<<endl;

n1.getdata();

cout<<"for number 2"<<endl;

n2.getdata();

n3=n1-n2;

cout<<"after n3=n1-n2"<<endl;

n3.putdata();

if(n1==n2)

{

cout<<"object are same";

}

else

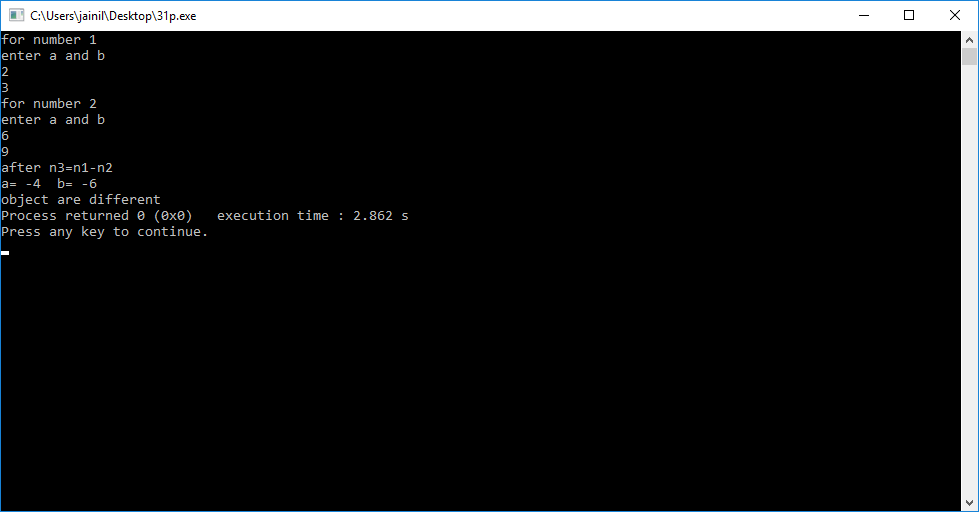
{

cout<<"object are different";

}

return 0;

}



**31. Create a class Measure having members: meter and cm. The class has get( ) and put( ) functions. Overload operator + and – such that they support M1=M2+15 and M3=M1 – 4.5. Also overload + and – such that they support M1=5.0+M2 and M3=2.0 – M4. Write a main( ) to test the class.**

**Use the concept of Overloading Binary Operators with friend function.**

/\*Create a class Measure having members: meter and cm. The class has

get( ) and put( ) functions. Overload operator + and – such that they

support M1=M2+15 and M3=M1 – 4.5. Also overload + and – such

that they support M1=5.0+M2 and M3=2.0 – M4. Write a main( ) to

test the class.

Use the concept of Overloading Binary Operators with friend

function.\*/

#include<iostream>

#include<cstdio>

using namespace std;

class measure

{

int metre;

int cm;

public:

void get()

{

cout<<"enter metre:";

cin>>metre;

cout<<"enter centimetre:";

cin>>cm;

start:

if(cm>100)

{

int x;

x=cm/100;

metre=metre+x;

cm=cm%100;

goto start;

}

else

{

cout<<"you have entered "<<metre<<" metre "<<cm<<" centimetre "<<endl;

}

}

void put()

{

cout<<metre<<" metre "<<cm<<" centimetre "<<endl;

}

friend void display(measure m1,measure m2,measure m3,measure m4);

friend measure operator +(measure m,int x);

friend measure operator -(measure m1,float f);

friend measure operator +(float f,measure m1);

friend measure operator -(float f,measure m1);

};

measure operator +(measure m,int x)

{

m.metre=m.metre+x;

return m;

}

measure operator -(measure m1,float f)

{

float z=m1.metre+((float)m1.cm/100);

float y=z-f;

cout<<y<<"m ";

measure ans;

ans.metre=(int)y;

ans.cm=(y-(int)y)\*100;

return ans;

}

measure operator +(float f,measure m1)

{

float z=m1.metre+((float)m1.cm/100);

float y=f+z;

cout<<y<<"m ";

measure ans;

ans.metre=(int)y;

ans.cm=(y-(int)y)\*100;

return ans;

}

measure operator -(float f,measure m1)

{

float z=m1.metre+((float)m1.cm/100);

float y=f-z;

cout<<y<<"m ";

measure ans;

ans.metre=(int)y;

ans.cm=(y-(int)y)\*100;

return ans;

}

void display(measure m1,measure m2,measure m3,measure m4)

{

cout<<"m1:"<<m1.metre<<"."<<m1.cm<<"m"<<endl;

cout<<"m2:"<<m2.metre<<"."<<m2.cm<<"m"<<endl;

cout<<"m3:"<<m3.metre<<"."<<m3.cm<<"m"<<endl;

cout<<"m4:"<<m4.metre<<"."<<m4.cm<<"m"<<endl;}

int main()

{ measure m1,m2,m3,m4;

cout<<" for m2"<<endl;

m2.get();

cout<<" for m4"<<endl;

m4.get();

m1=m2+15;

cout<<" for m1=m2+15"<<endl;

m1.put();

m3=m1-4.563;

cout<<" for m3=m1-4.5"<<endl;

m3.put();

m1=5.0+m2;

cout<<" for m1=5.0+m2"<<endl;

m1.put();

m3=2.0-m4;

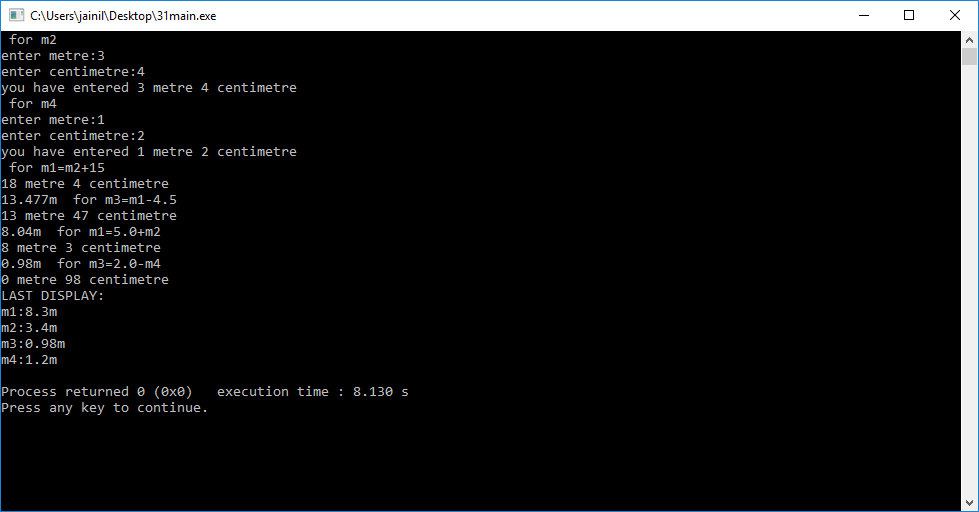
cout<<" for m3=2.0-m4"<<endl;

m3.put();

cout<<"LAST DISPLAY:"<<endl;

display(m1,m2,m3,m4);

return 0;}



**32. Create a class Money having integer rupee, paisa as data. Define appropriate functions to enter and display data. Also define function that supports the statement: obj1=115. Here obj1 is object of class and 115 is integer that represents paisa. After execution of the statement obj1 will be 1 rupee 15 paisa.**

**Use the concept of Type conversion from basic type to class type.**

#include<iostream>

#include<cstdio>

using namespace std;

class money

{

int rs;

int p;

public:

money(){};

money(int x)

{

rs=x/100;

p=x%100;

}

void putdata()

{

cout<<rs<<" rupee "<<p<<" paisa ";

}

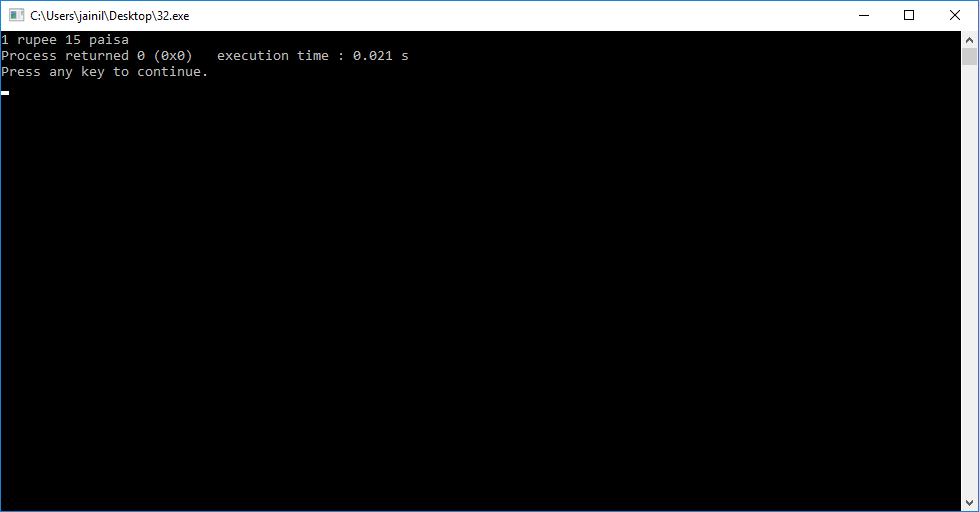
};

int main()

{ money m1;

m1=115;

m1.putdata();}



**33. Create a class Celsius with float. Define appropriate member functions such that it support the statements:  float temperature; temperature=C2;Use the concept of Type conversion from class type to basic type.**

#include<iostream>

#include<cstdio>

using namespace std;

class celsius

{

float temp;

public:

celsius ()

{

}

celsius(float x)

{

temp=x;

}

operator float()

{ return temp; }

};

int main()

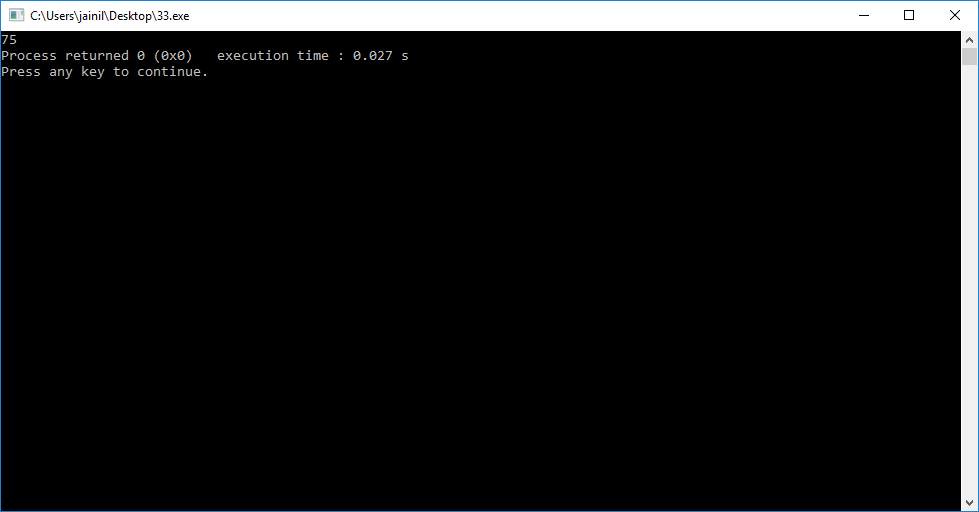
{ celsius t;

float temperature;

t=75;

temperature=t;

cout<<temperature;}



**34) Define a class KG having data member float kg and class POUND having data member float lb. Both the classes have default constructor and member functions to get and show data. (1 kg = 2.20462 lb). Define appropriate memberfunctions such that they support the statements in main( ):**

**KG K;**

**POUND P;**

**P=K;**

**Use the concepts of Type conversion from class type to class type.**

**Write this Program in two ways.**

**Define appropriate member function in class KG.**

**Define appropriate member function in class POUND**

#include<iostream>

#include<cstdio>

using namespace std;

class POUND

{

float lb;

public:

POUND(){}

POUND(float k)

{

lb=k;

}

//POUND(KG k)

//{

// lb=k.getfloat()\*2.20462;

//}

void getdata()

{

cout<<"enter pound";

cin>>lb;

}

void putdata()

{

cout<<"pound is :"<<lb<<" lb";

}

};

class KG

{

float kg;

public:

float getfloat()

{

return kg;

}

void getdata()

{

cout<<"enter kg:";

cin>>kg;

}

void putdata()

{

cout<<"kilogram is :"<<kg<<"kg";

}

operator POUND()

{

return POUND(kg\*2.20462);

}

};

int main()

{

cout<<"1 KG = 2.20462"<<endl;

POUND p;

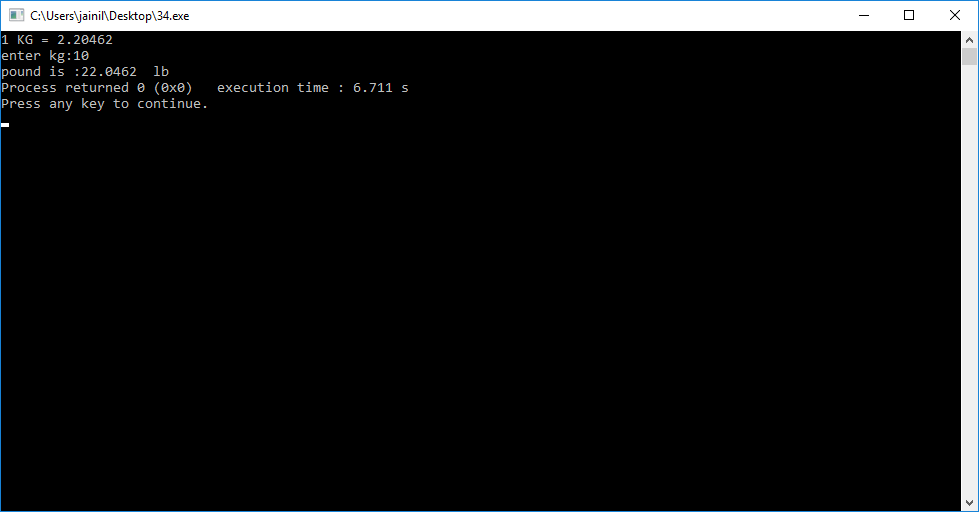
KG k;

k.getdata();

p=k;

p.putdata();

}



**35) Define a Base Class Vegetable having data member Color and member function getdata() which takes color as an input and putdata() which print the color as an output.**

**Vegetable Class has one subclass named Tomato having data members weight and size and member function gtdata() which takes weight and size as an input and ptdata()**

**which prints weight and size as output. Write a C++ Program which inherits the data of Vegetable class in Tomato class using Single Inheritance.**

#include<iostream>

using namespace std;

class vegetable

{

string color;

public:

int x;

void getdata()

{

cout<<"enter colour of vegetable:";

cin>>color;

}

void putdata()

{

cout<<color<<endl;

}

};

class tomato:public vegetable

{

float weight;

float sizee;

public:

int x;

void gtdata()

{

cout<<"enter size of tomato:";

cin>>sizee;

cout<<"enter weight of tomato:";

cin>>weight;

}

void ptdata()

{

cout<<"size:"<<sizee<<endl;

cout<<"weight"<<weight<<endl;

}

};

int main()

{

tomato t;

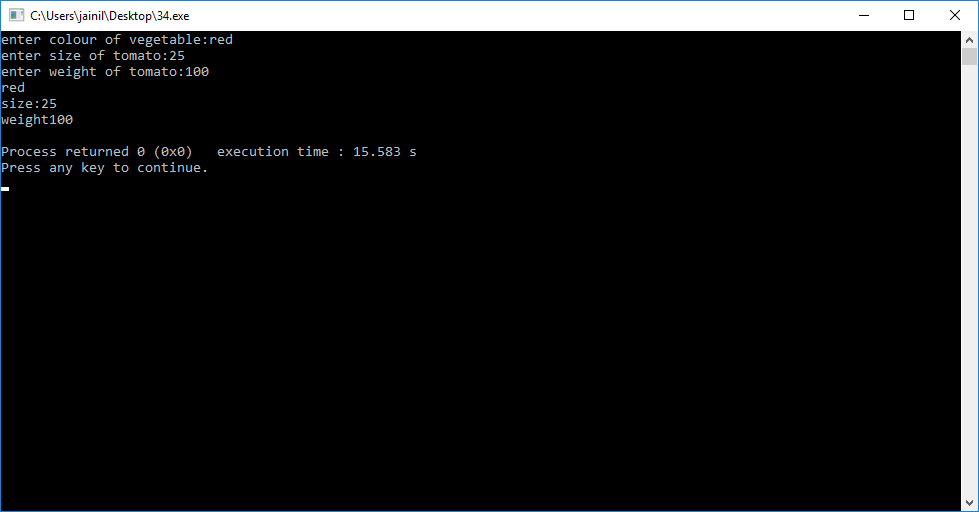
t.getdata();

t.gtdata();

t.putdata();

t.ptdata();

}



**36) Write a program to create a class Medicine which stores type of medicine, name of company, date of manufacturing. Class Tablet is inherited from Medicine. Tablet class has name of tablet, quantity per pack, price of one tablet as members. Class Syrup is also inherited from Medicine and it has quantity per bottle, dosage unit as members. Both the classes contain necessary member functions for input and output data. Write a main( ) that enter data for tablet and syrup, also display the data. Use the concepts of Hierarchical Inheritance.**

#include<iostream>

#include<cstring>

using namespace std;

class medicine

{

string type;

string nameofmanufacture;

int dd,mm,yyyy;

public:

void getdatam()

{

cout<<"enter type of medicine:";

cin>>type;

cout<<"enter name of manufacture:";

cin>>nameofmanufacture;

cout<<"enter date of manufacture in dd mm yyyy format:";

cin>>dd>>mm>>yyyy;

}

void putdatam()

{

cout<<type<<endl<<nameofmanufacture<<endl<<dd<<":"<<mm<<":"<<yyyy<<endl;

}

string t()

{

return type;

}

};

class tablet:public medicine

{

string nameoftablet;

int quantityperpack;

int priceofonetablet;

public:

void getdatat()

{

cout<<"enter name of tablet:";

cin>>nameoftablet;

cout<<"enter quantity per pack:";

cin>>quantityperpack;

cout<<"enter price of one tablet:";

cin>>priceofonetablet;

}

void putdatat()

{

cout<<nameoftablet<<endl<<quantityperpack<<endl<<priceofonetablet<<endl;

}

};

class syrup:public medicine

{

int quantityperbottle;

int dosageunit;

public:

void getdatas()

{

cout<<"enter quantity per bottle:";

cin>>quantityperbottle;

cout<<"enter dosage unit:";

cin>>dosageunit;

}

void putdatas()

{ cout<<quantityperbottle<<endl<<dosageunit<<endl;}

};

int main()

{

medicine m;

string type;

cout<<"enter type:";

cin>>type;

if(type=="tablet")

{

tablet t;

t.getdatam();

t.getdatat();

t.putdatam();

t.putdatat();

}

else if(type=="syrup")

{

syrup s;

s.getdatam();

s.getdatas();

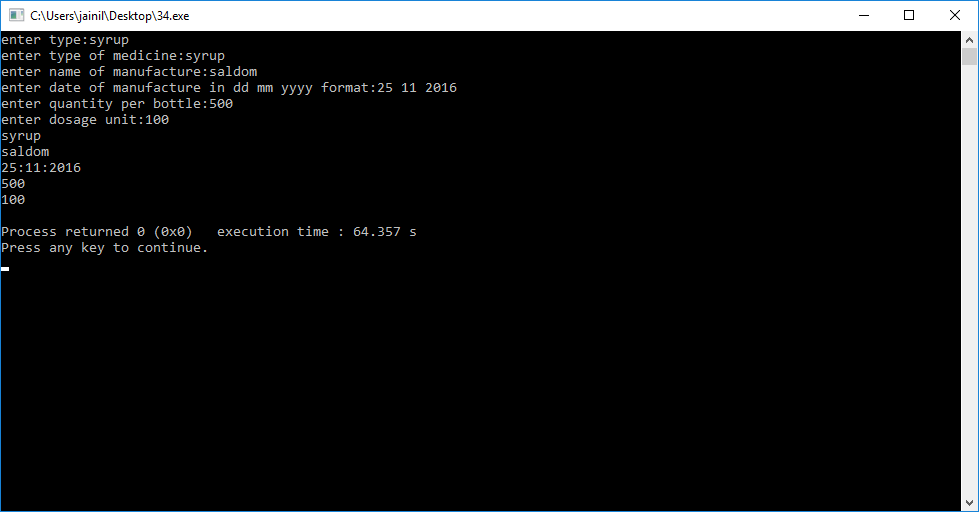
s.putdatam();

s.putdatas();

}

return 0;

}



**37) Create a class shape having data member shape\_name and member function to get and print shape\_name. Derive a Class Circle which is inherited publicly from class shape and having data members radius of a circle and member function to get and print radius of a circle. Derive a Class Area which is inherited publicly from Class Circle and having data members area\_of\_circle and member function display () which displays area of a circle. Use object of class Area in main () function and get and display all the information. Use the concepts of Multilevel Inheritance.**

#include<iostream>

using namespace std;

class shape

{

string shapename;

public:

void getdatas()

{

cout<<"enter shape:";

cin>>shapename;

}

void putdatas()

{

cout<<shapename<<endl;

}

};

class circle:public shape

{

public:

float radius;

void getr()

{

cout<<"enter radius:";

cin>>radius;

}

void putr()

{

cout<<radius<<endl;

}

};

class area:public circle

{

float area;

public:

void display()

{

area=radius\*radius\*3.14;

cout<<"area="<<area<<endl;;

}

};

int main()

{

area a;

a.getdatas();

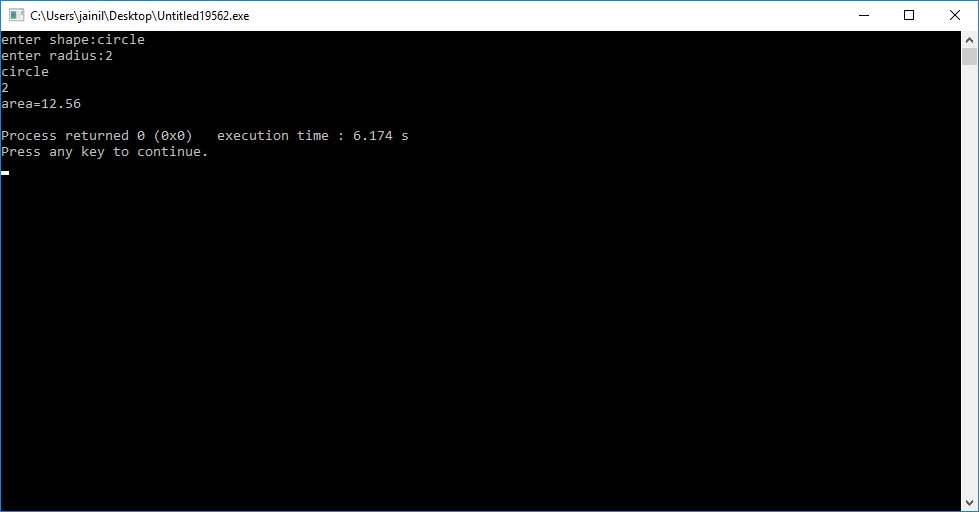
a.getr();

a.putdatas();

a.putr();

a.display();

}



**38) Define a class Hospital having rollno and name as data members and member function to get and print data. Derive a class Ward from class Hospital having data members: ward number and member function to get and print data. Derive another class Room from Hospital having data member bed number and nature of illness and member function to get and print data. Derive class Patient from Class Ward and Class Room. In main () declare 5 object of Class Patient and get and display all the information.**

**Use the concept of Virtual Base Class and Hybrid Inheritance.**

#include<iostream>

using namespace std;

class hospital

{

int rollno;

public:

string name;

void getdatah()

{

cout<<"enter roll no:";

cin>>rollno;

cout<<"enter name of patient:";

cin>>name;

}

void putdatah()

{

cout<<"roll no:"<<rollno<<endl<<"name of patient"<<name<<endl;

}

};

class ward:virtual public hospital

{

int wardno;

public:

void getdataw()

{

cout<<"enter ward no:";

cin>>wardno;

}

void putdataw()

{

cout<<"ward no is:"<<wardno<<endl;

}

};

class room:virtual public hospital

{

int bednumber;

public:

string nature;

void getdatar()

{

cout<<"enter bed no:";

cin>>bednumber;

cout<<"enter nature of illness:";

cin>>nature;

}

void putdatar()

{

cout<<"bed no:"<<bednumber<<endl;

cout<<"nature of illness"<<nature<<endl;

}

};

class patient:public ward,public room

{

};

int main()

{

patient p[5];

for(int i=0;i<5;i++)

{

cout<<"for "<<i+1<<" patient"<<endl;

p[i].getdatah();

p[i].getdataw();

p[i].getdatar();

}

for(int i=0;i<5;i++)

{

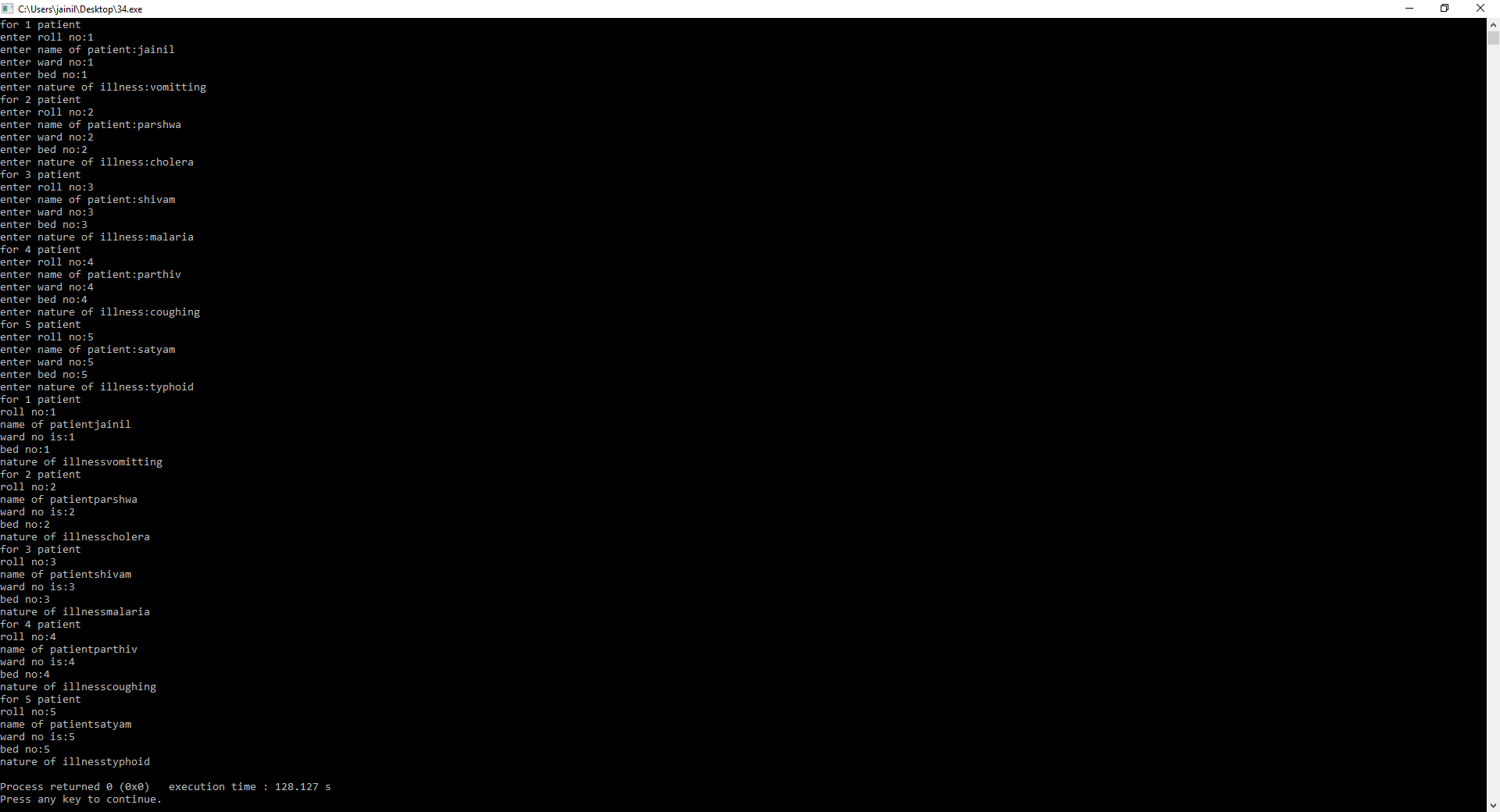
cout<<"for "<<i+1<<" patient"<<endl;

p[i].putdatah();

p[i].putdataw();

p[i].putdatar();

}}



**39) Create a Class alpha having data member: int x and one argument constructor which initializes the value of x. It also has member function which displays the value of x. Create another class beta which contains data member: float y and one argument constructor which initializes the value of y. It also has member function which displays the value of y. Create a Class Gamma which publicly inherits from class alpha and class beta and has two data members: int m, n and a constructor which passes argument to the base class constructor as well as initializes its own data members. Class Gamma also has member function to print the values of m and n. Write main function which creates object of class Gamma which passes values of base class constructor as well as derived class constructor. Use the concept of Multiple Inheritance and Constructor in Derived Class.**

#include<iostream>

using namespace std;

class alpha

{

int x;

public:

alpha (int a)

{

x=a;

}

void displayx()

{ cout<<"x= "<<x<<endl; }

};

class beta

{

float y;

public:

beta (float a)

{

y=a;

}

void displayy()

{

cout<<"y= "<<y<<endl;

}

};

class gamma:public alpha,public beta

{

int m;int n;

public:

gamma(int a,float b,int c,int d):alpha(a),beta(b)

{

m=c;n=d;

}

void displaymn()

{

cout<<"m= "<<m<<endl<<"n= "<<n<<endl;

}

};

int main()

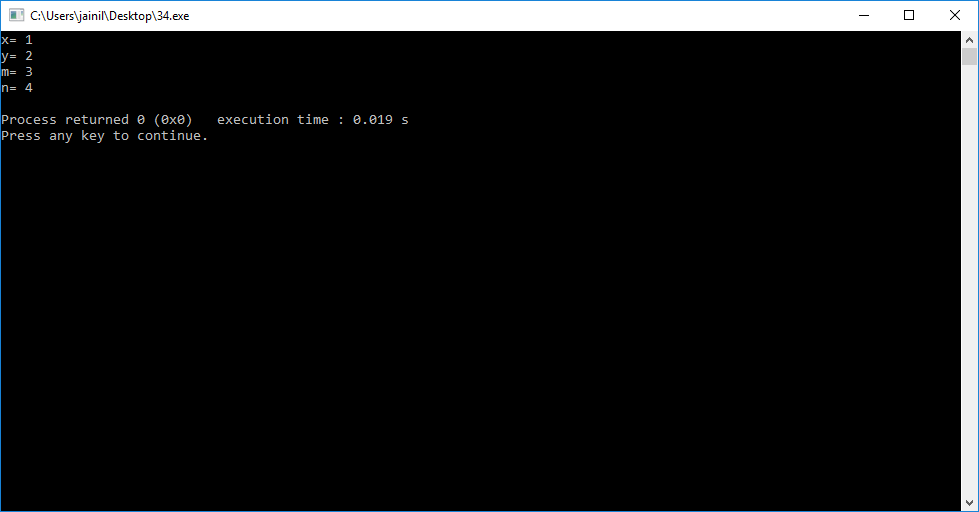
{ gamma c(1,2,3,4);

c.displayx();

c.displayy();

c.displaymn();

return 0;

}

**40) What is the output of the following code:**

* **Pointer to Objects**

**#include<iostream>**

**using namespace std;**

**class product**

**{**

**int code;**

**float price;**

**public:**

**void getdata(int a, float b)**

**{**

**code=a;**

**price=b;**

**}**

**void show()**

**{**

**cout<<"Code: "<<code<<endl;**

**cout<<"Price: "<<price<<endl;**

**}**

**};**

**int main()**

**{**

**product \* p = new product;**

**product \*d = p;**

**int x,i;**

**float y;**

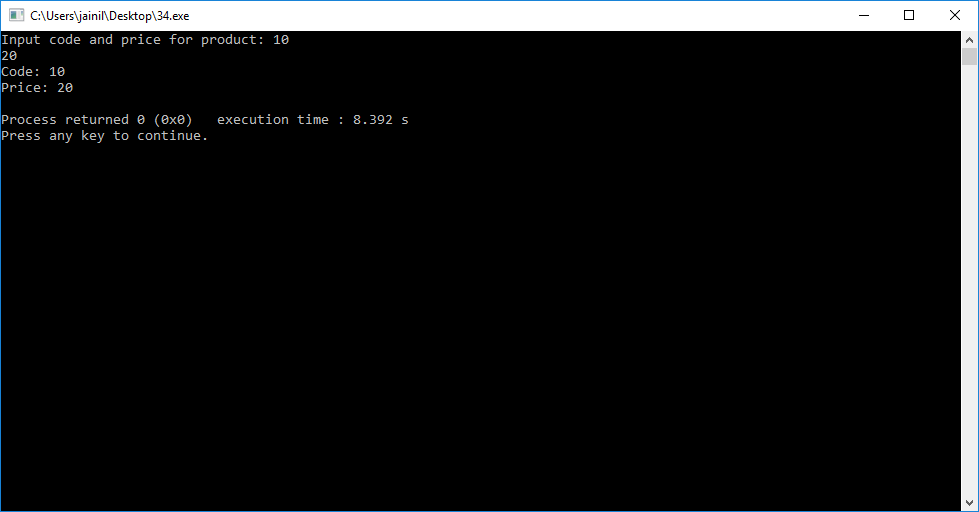
**cout<<"Input code and price for product: ";**

**cin>>x>>y;**

**p->getdata(x,y);**

**d->show();**

**}**



* **this pointer**

**#include<iostream>**

**using namespace std;**

**class student**

**{**

**int roll\_no;**

**float age;**

**public:**

**student(int r, float a)**

**{**

**roll\_no = r;**

**age = a;**

**}**

**student & greater (student & x)**

**{**

**if(x.age>=age)**

**return x;**

**else**

**return \*this;**

**}**

**void display()**

**{**

**cout<<"Roll No "<<roll\_no<<endl;**

**cout<<"Age "<<age<<endl;**

**}**

**};**

**int main()**

**{**

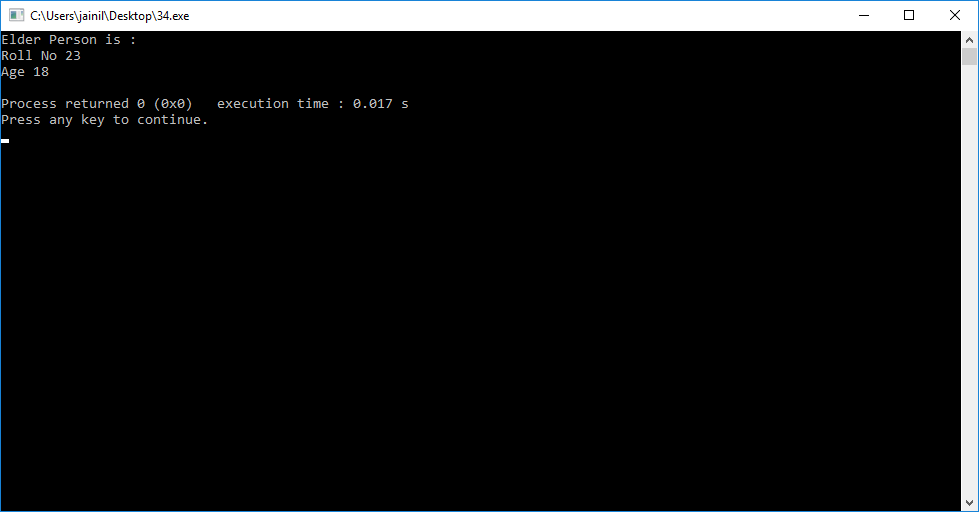
**student s1 (23,18),s2 (30,20),s3 (45,16);**

**student s = s1.greater(s3);**

**cout<<"Elder Person is :"<<endl;**

**s.display();**

**}**



* **Pointers to Derived Objects**

**#include<iostream>**

**using namespace std;**

**class BC**

**{**

**public:**

**int b;**

**void show()**

**{**

**cout<<"b = "<<b<<endl;**

**}**

**};**

**class DC : public BC**

**{**

**public:**

**int d;**

**void show()**

**{**

**cout<<"b = "<<b<<endl;**

**cout<<"d = "<<d<<endl;**

**}**

**};**

**int main()**

**{**

**BC \*bptr;**

**BC base;**

**bptr = &base;**

**bptr->b = 100;**

**cout<<"bptr poins to base objects"<<endl;**

**bptr->show();**

**DC derived;**

**bptr = &derived;**

**bptr->b = 200;**

**/\*bptr->b = 300;\*/ // wont work**

**cout<<"bptr now points to derived object"<<endl;**

**bptr->show();**

**DC \*dptr;**

**dptr=&derived;**

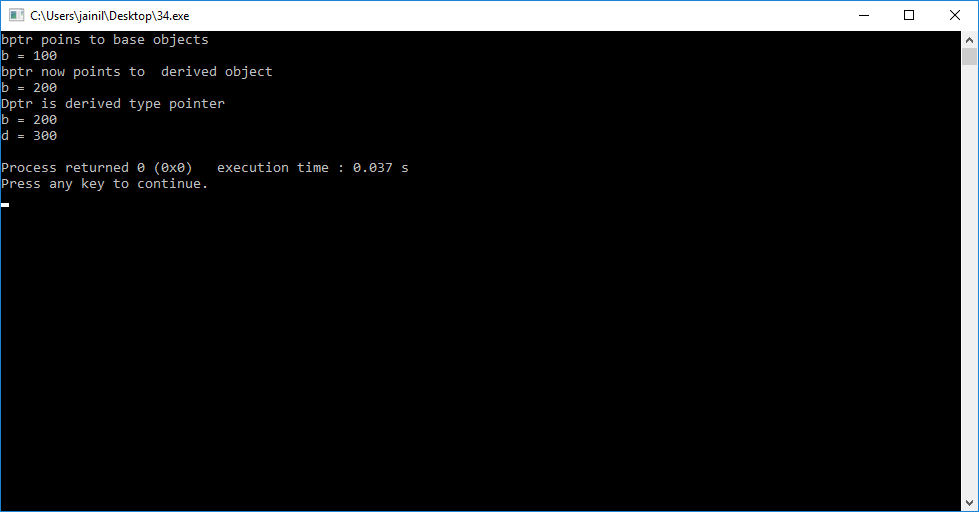
**dptr->d=300;**

**cout<<"Dptr is derived type pointer"<<endl;**

**dptr->show();**

**return 0;**

**}**



**41) Create a class Media that stores the title (a string) and price (float). Class Media has two argument constructor which initializes data members of class Media. Also declare a virtual function display () in Class Media. From the class Media derive two classes: Class book, which contains data member page count (int): and Class tape, which contains data member playing time in minutes (float). Both Class book and Class tape should have a constructor which initializes base class constructor as well as its own data members and display ( ) function which displays book details and tape details respectively. Write a main ( ) to test book and tape classes by creating instances of them, asking the user to fill data and displaying them. Use the concept of Virtual function and Constructor in Derived Class. Virtual function is also known as Runtime Polymorphism or Dynamic Binding.**

#include<iostream>

using namespace std;

class media

{

protected:

string title;

float price;

public:

media(){}

media(string a,float b)

{

title=a;

price=b;

}

virtual void display()

{

cout<<title<<" "<<price;

}

;

};

class book:public media

{

int pagecount;

public:

book(){}

book(string a,float b,int c):media(a,b)

{

pagecount=c;

}

void display()

{

cout<<"title:"<<endl<<title<<endl<<"price:"<<price<<endl<<"pagecount="<<pagecount<<endl;

}

};

class tape:public media

{

float minute;

public:

tape(){}

tape(string a,float b,float c):media(a,b)

{

minute=c;

}

void display()

{

cout<<"title:"<<title<<endl<<"price:"<<price<<endl<<"time in min:"<<minute<<endl;

}

};

int main()

{

string a;

float b;

int c;

float d;

cout<<"for BOOKS:"<<endl;

cout<<"enter title of book:"<<endl;

cin>>a;

cout<<"enter price of book:"<<endl;

cin>>b;

cout<<"enter page count:"<<endl;

cin>>c;

book e(a,b,c);

media \*ptr=&e;

ptr->display();

cout<<endl<<endl<<endl<<endl<<endl;

cout<<"for TAPE:"<<endl;

cout<<"enter title of tape:"<<endl;

cin>>a;

cout<<"enter price of tape:"<<endl;

cin>>b;

cout<<"enter time in minutes:"<<endl;

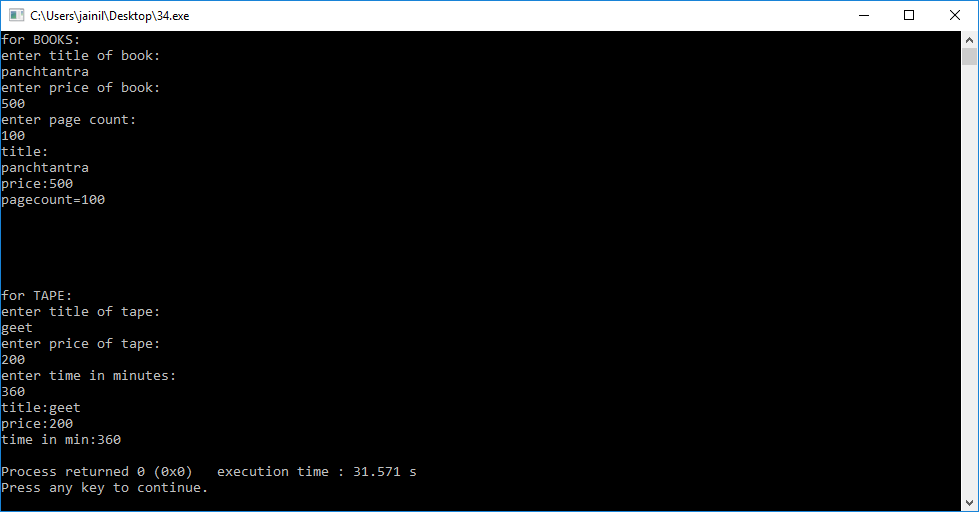
cin>>d;

tape t(a,b,d);

ptr=&t;

ptr->display();

}



**42) Create a Abstract class vehicle having average as data and pure virtual function getdata() and putdata(). Derive class car and truck from class vehicle having data members: fuel type (petrol, diesel, CNG) and no of wheels respectively. Write a main ( ) that enters the data of two cars and a truck and display the details of them. Use the concept of Abstract Base class and Pure Virtual functions.**

#include<iostream>

using namespace std;

class vehicle

{

public:

float average;

virtual void getdata()=0;

virtual void putdata()=0;

};

class car:public vehicle

{

string fuel;

int noofwheels;

public:

void getdata()

{

cout<<"enter average of car"<<endl;

cin>>average;

cout<<"enter fuel type of car"<<endl;

cin>>fuel;

cout<<"enter number of wheels"<<endl;

cin>>noofwheels;

}

void putdata()

{

cout<<"average of car="<<average<<endl<<"fuel of car="<<fuel<<endl<<"no of wheels of car"<<noofwheels<<endl;

}

};

class truck:public vehicle

{

string fuel;

int noofwheels;

public:

void getdata()

{

cout<<"enter average of truck"<<endl;

cin>>average;

cout<<"enter fuel type of truck"<<endl;

cin>>fuel;

cout<<"enter number of truck"<<endl;

cin>>noofwheels;

}

void putdata()

{

cout<<"average of truck="<<average<<endl<<"fuel of truck="<<fuel<<endl<<"no of wheels of truck"<<noofwheels<<endl;

}

};

int main()

{

truck t;

car c;

car d;

c.getdata();

c.putdata();

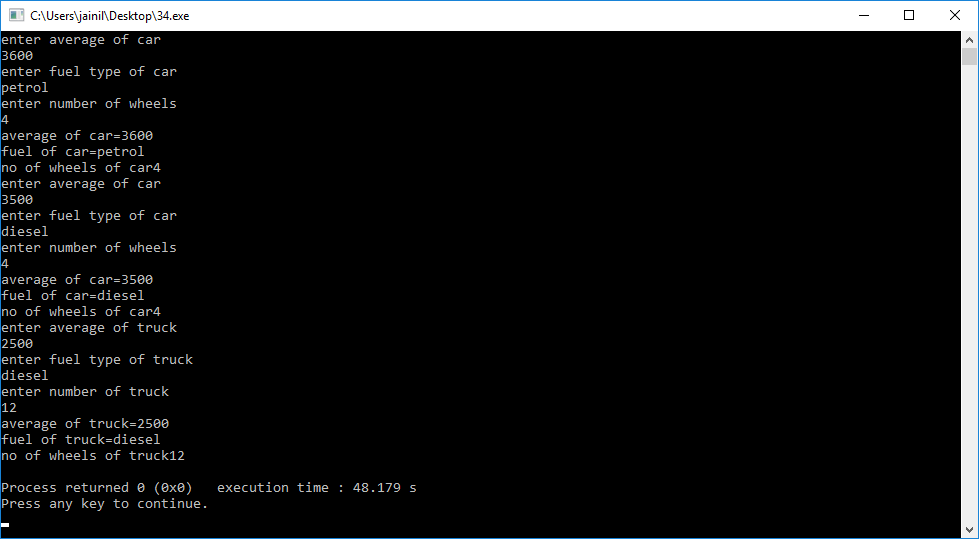
d.getdata();

d.putdata();

t.getdata();

t.putdata();

}



**43.) What is the output of the following code related to ios format functions, get() and put() functions and getline() and write() functions?**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**char s[12]="ABC\_DEF\_GHI";**

**cout.write(s,9);**

**int x=12345;**

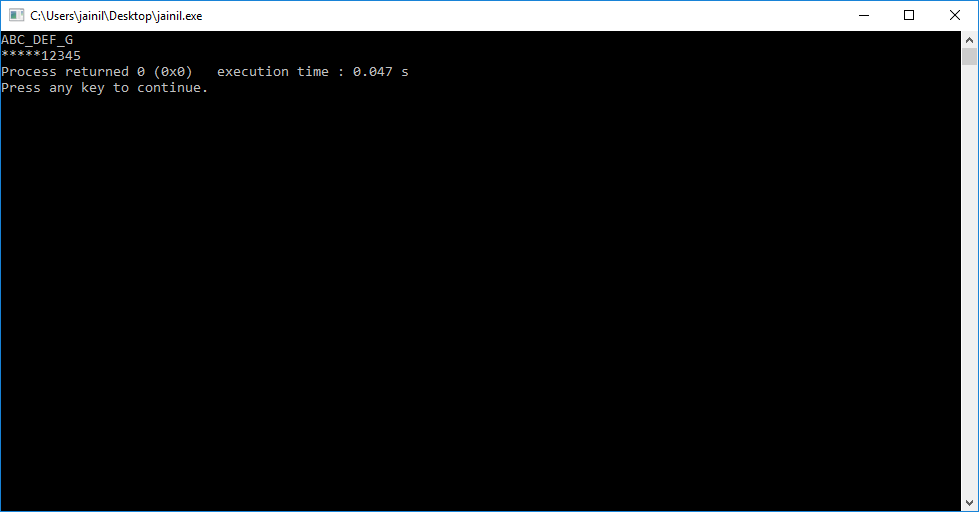
**cout.fill('\*');**

**cout.width(10);**

**cout<<endl<<x;**

**return 0;**

**}**



**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int a,b;**

**a = (b = 50) +10;**

**cout<<"a = "<<a<<endl; cout<<"b = "<<b<<endl;**

**float x=23.4;**

**cout.fill('\*');**

**cout.width(10);**

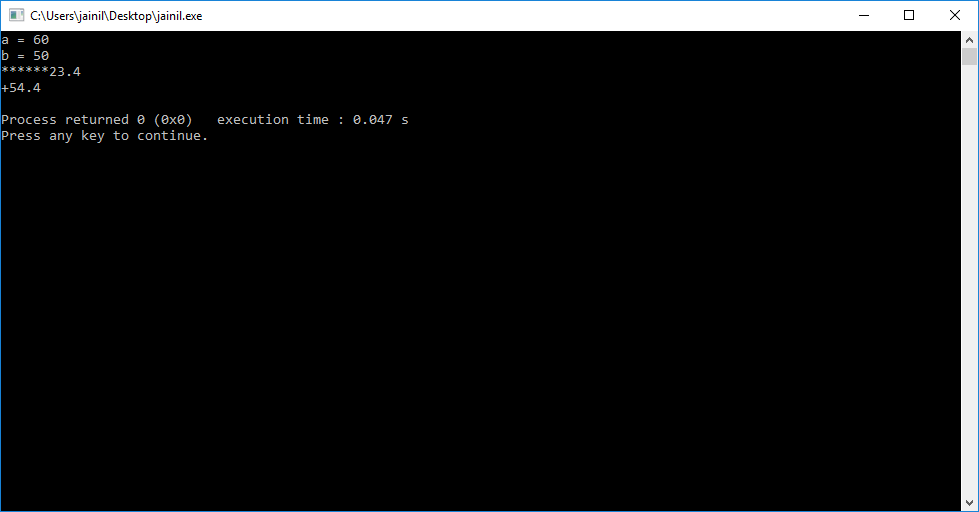
**cout<<x<<endl;**

**float y=54.4;**

**cout.setf(ios::showpos);**

**cout<<y<<endl;**

**return 0; }**



**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int count =0;**

**char c;**

**cout<<"INPUT TEXT\n";**

**cin.get(c);**

**while(c!='\n')**

**{**

**cout.put(c);**

**count++;**

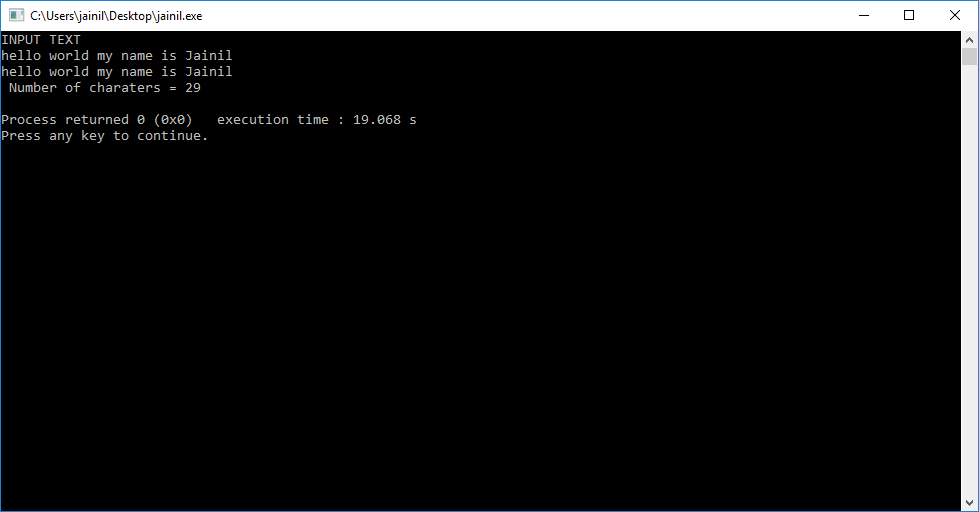
**cin.get(c);**

**}**

**cout<<"\n Number of charaters = "<<count<<"\n";**

**return 0;**

**}**



**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**char name[20];**

**cout<<"Enter first name then white space and then last name of a person: ";**

**cin>>name;**

**cout<<"Person Name : "<<name<<endl;**

**cout<<"Enter first name then white space and then last name of a person: ";**

**cin.getline(name,10);**

**cout.write(name,7);**

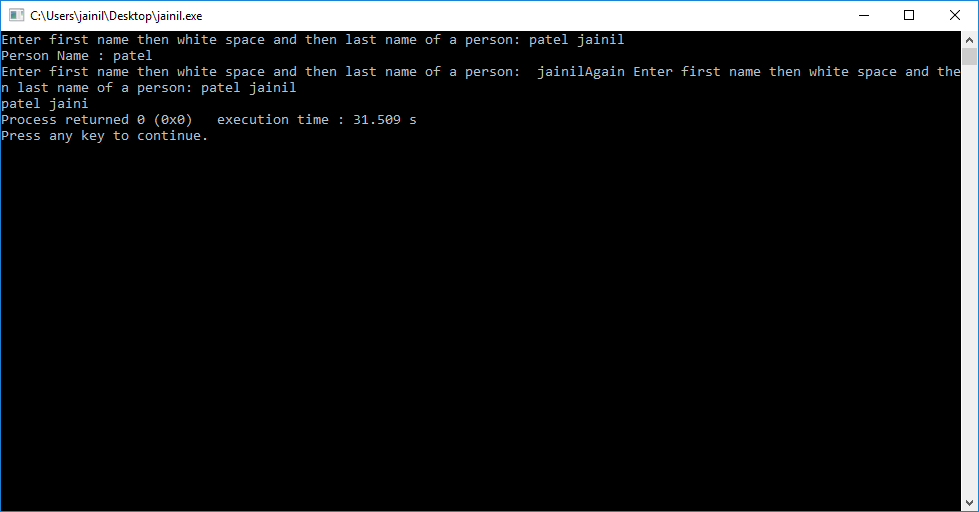
**cout<<"Again Enter first name then white space and then last name of a person: ";**

**cin.getline(name,13);**

**cout.write(name,11);**

**return 0;**

}



**44.) Write a program which demonstrates how to create user-defined Manipulators.**

#include<iostream>

#include<iomanip>

using namespace std;

ostream & currency(ostream & output)

{

output<<"Rs";

return output;

}

ostream &form(ostream &output)

{

output.setf(ios::showpos);

output.setf(ios::showpoint);

output.fill('\*');

output.precision(2);

output<<setiosflags(ios::fixed)<<setw(10);

return output;

}

int main()

{

float no;

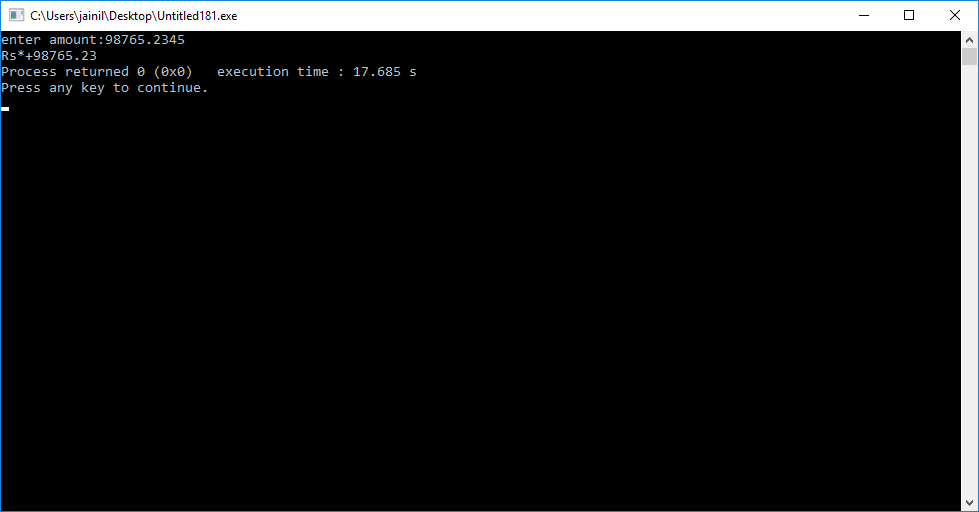
cout<<"enter amount:";

cin>>no;

cout<<currency<<form<<no;

return 0;

}



**45.) Write a program that creates a file, insert name of a person in the file and copy the contents in another file. Use concept of Opening the file using constructor. Use all error handling functions like eof() , fail() , bad() and good() functions.**

#include<iostream>

#include<fstream>

using namespace std;

int main()

{ string name;

ofstream file1("name.txt");

cout<<"enter name:";

cin>>name;

file1<<name;

file1.close();

ifstream file3("name.txt");

ofstream file2("duplicate.txt");

char l[30];

{ file3.getline(l,30);

file2<<l;}

file2.close();

ifstream file4("duplicate.txt");

char k[30];

file4.getline(k,30);

cout<<k;

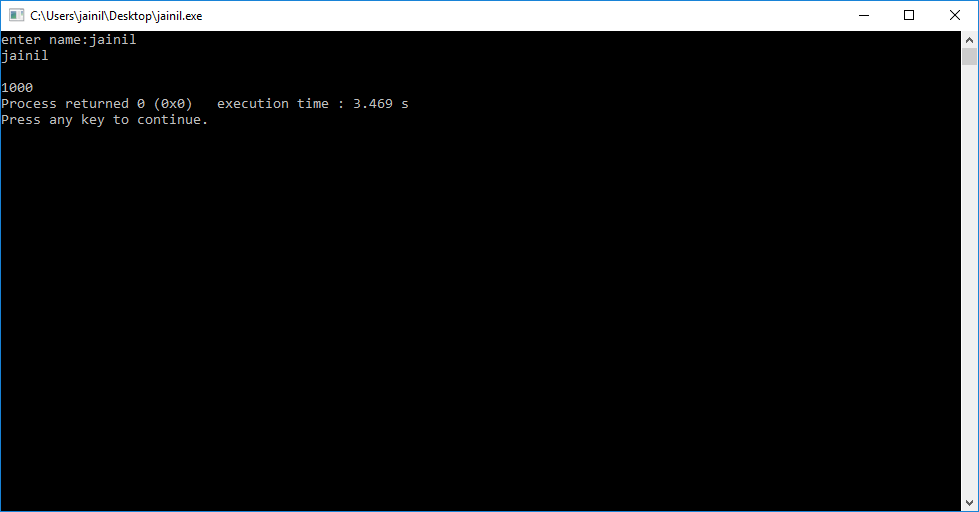
cout<<"\n\n";

cout<<file4.eof();

cout<<file4.fail();

cout<<file4.bad();

cout<<file4.good();}



**46.) Write a program that creates a text file that contains ABC…Z. A program should print the file in reverse order on the screen. i.e. ZYX…BA. Use concept of Opening the file using open () functions and functions for manipulation of file pointer like seekg() and tellg().**

#include<iostream>

#include<fstream>

using namespace std;

int main()

{ char a='A';

ofstream abc("abc.txt");

for(int i=0;i<26;i++)

{ abc<<a; a++;}

abc.close();

ifstream file1("abc.txt");

ofstream file2("cba.txt");

char b[1];

file1.seekg(1,ios::end);

for(int i=0;i<26;i++)

{ char ch;

file1.seekg(-2L,ios::cur);

file1.get(ch);

file2<<ch;}

file2.close();

file1.close();

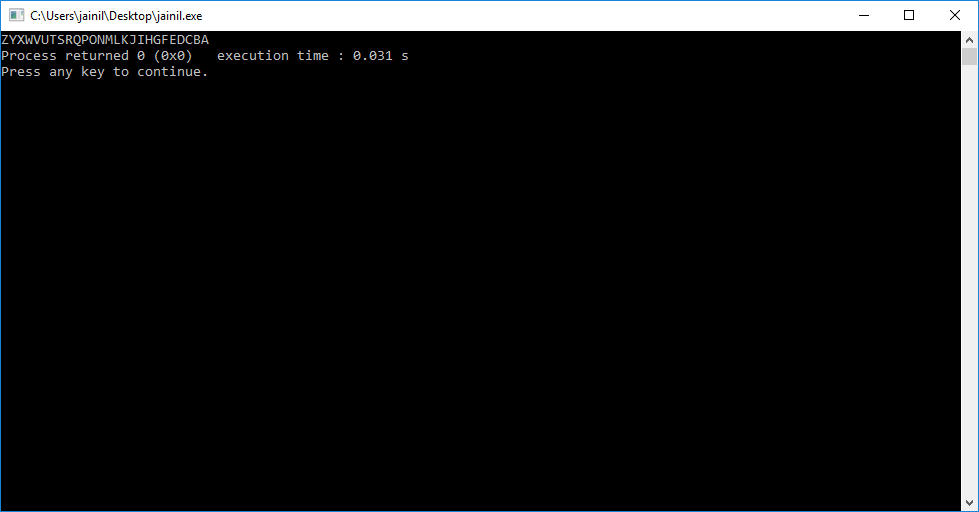
ifstream file4("abc.txt");

ifstream file3("cba.txt");

char p[100];

file3.getline(p,100);

cout<<p;}



**47.) Write a program that creates a binary file and input height in float for the five students. Display the content of the file with two precision. Use the concept of Write() and read() functions for handling data in binary form.**

#include<iostream>

#include<fstream>

using namespace std;

int main()

{

float g[5];

for(int i=0;i<5;i++)

{

cout<<"enter "<<i+1<<" "<<"height:";

cin>>g[i];

}

ofstream f("123.txt");

f.write((char \*)&g,sizeof(g));

f.close();

ifstream a("123.txt");

float s[5];

a.read((char \*)&s,sizeof(s));

a.close();

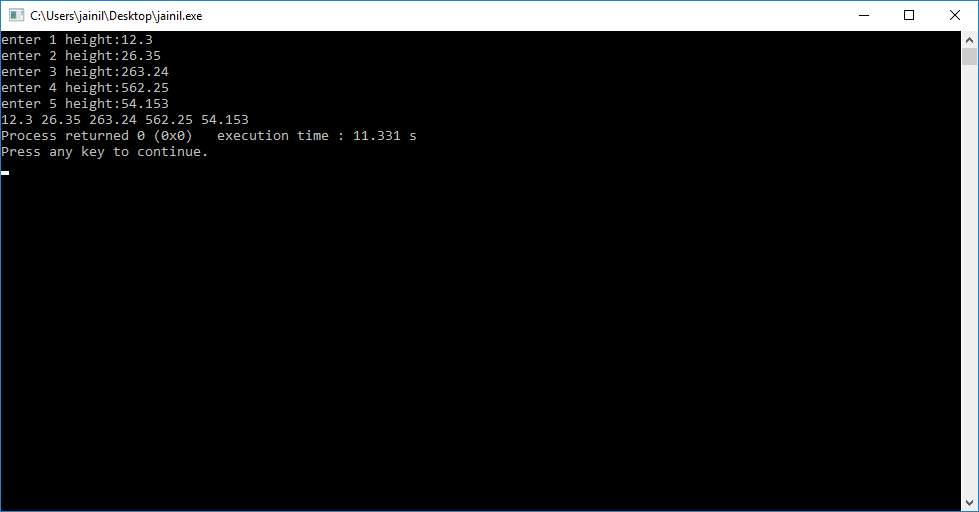
for(int i=0;i<5;i++)

{

cout<<s[i]<<" ";

}

}



**48.) Write a program using command-line arguments that copy a source into destination file. c:>copyfile a.txt b.txt.**

#include<iostream>

#include<fstream>

using namespace std;

int main(int argc,char \* argv[])

{

char a[1000];

cout<<"source file opened :"<<argv[1];

ofstream sour(argv[1]);

cout<<"\n"<<"enter data in file"<<"\n";

char ch='a';

while(ch!=EOF)

{

ch=getchar();

sour.put(ch);

}

sour.close();

ifstream sour1(argv[1]);

ofstream dest1(argv[2]);

cout<<"copying starts to destination file \n";

ch='a';

int i=0;

while(sour1)

{

sour1.seekg(i,ios::beg);

sour1.get(ch);

dest1<<ch;

i++;

}

sour1.close();

dest1.close();

cout<<" \n printing copied data \n";

ifstream dest2(argv[2]);

ch='a';

i=0;

while(!dest2.eof())

{

dest2.seekg(i,ios::beg);

dest2.get(ch);

cout<<ch;

i++;

}

cout<<"\n ending program type a character:";

cin>>ch;

}

