**PROGRAM NO. 1**

Enter any string and sort it. (Use string class for string handling)

Source Code:

#include<iostream>

#include<algorithm>

using namespace std;

void sort(string &str)

{

sort(str.begin(), str.end());

cout<< str;

}

int main()

{

string a;

cout<<"Enter any String : ";

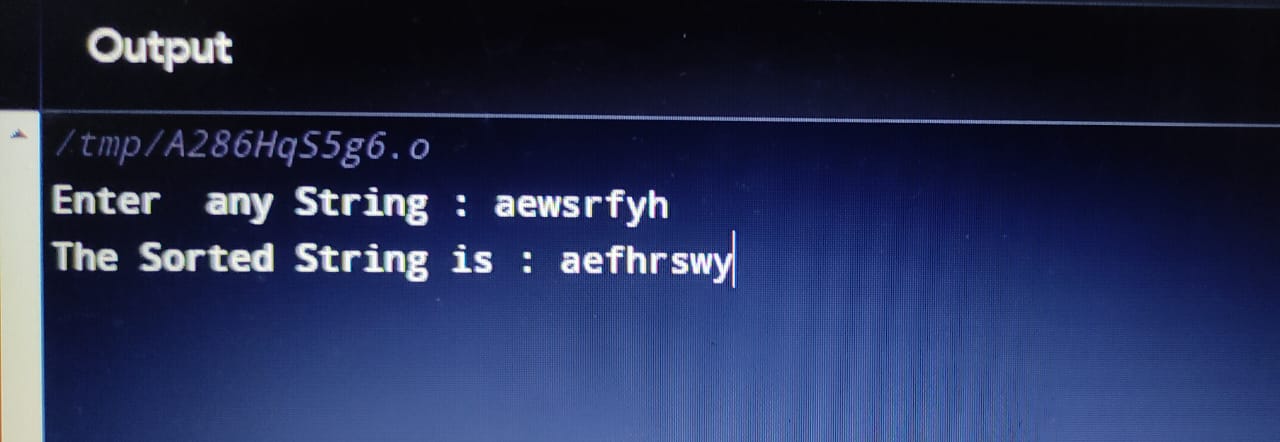
cin>>a;

cout<<"The Sorted String is : ";

sort(a);

}

**Output --:**



**PROGRAM NO. 2**

Create an array of ten names and sort them. (Use string class for string handling)

Source Code:

#include <iostream>

#include<algorithm>

using namespace std;

int main()

{

string s[] = {"xyz", "abc", "def", "pqr", "tuv", "cpp", "yes", "no", "hello"};

sort(s, s + 9);

for (int i = 0; i < 9; ++i)

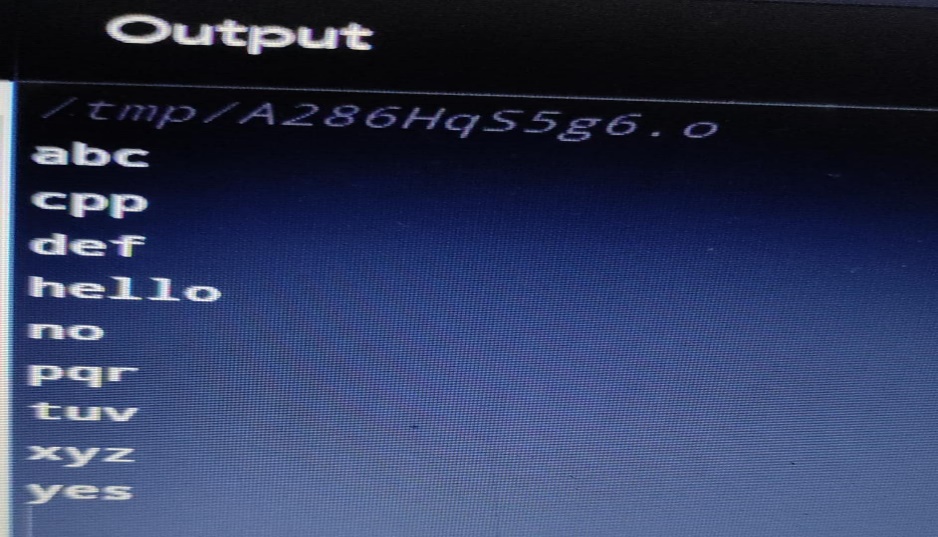
{

cout << s[i] << endl;

}

}

**Output --:**



**PROGRAM NO. 3**

Enter 10 numbers and do the following operations:

(i) sorting (ii) searching

(iii) deleting (iv) updating

Source Code:

#include <iostream>

using namespace std;

int arr[100];

void update(int val, int in){

arr[in] = val;

}

void del(int in, int \*n){

for (int i = in; i < \*n; i++){

arr[i] = arr[i + 1];

}

\*n = \*n - 1;

}

void search(int val, int \*n){

for (int i = 0; i < \*n; i++){

if (arr[i] == val){

cout << "Element Found at index : " << i << endl;

}

}

}

void sort(int \*n){

for (int i = 0; i < \*n - 1; i++){

for (int j = 0; j < \*n - 1 - i; j++){

if (arr[j + 1] < arr[j]){

int temp = arr[j + 1];

arr[j + 1] = arr[j];

arr[j] = temp;

}

}

}

}

void display(int \*n){

for (int i = 0; i < \*n; i++){

cout << arr[i] << " ";

}

cout << endl;

}

int main(){

int n, z, val, in;

cout << "Enter Size of Array";

cin >> n;

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

cin >> arr[i];

}

cout << "Choose\n";

while (1){

cout << "1. Update\n2. Delete\n3. Search\n4. Sorting\n5. Display\n6. Exit\n";

cin >> z;

switch (z){

case 1:

cin >> val >> in;

update(val, in);

break;

case 2:

cin >> in;

del(in, &n);

cout<<"Deleted";

break;

case 3:

cout<<"Enter Element to search : ";

cin >> val;

search(val, &n);

break;

case 4:

sort(&n);

cout<<"Array Sorted";

break;

case 5:

display(&n);

break;

case 6:

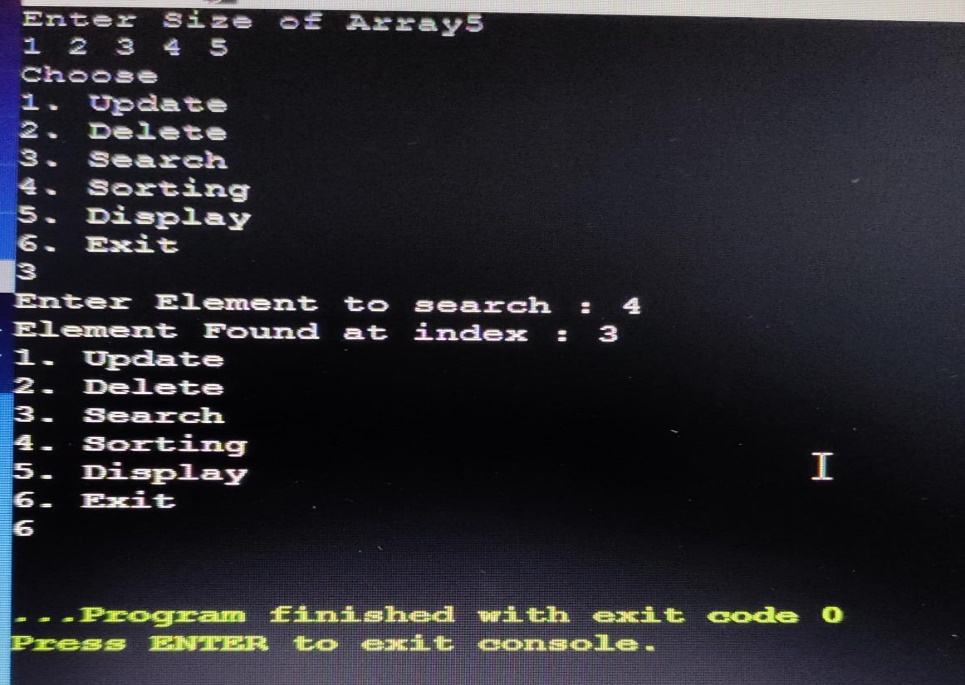
exit(0);

}

}

}

**Output --:**



**PROGRAM NO. 4**

Write a program to check whether the given program is palindrome?

Source Code:

#include<iostream>

#include<algorithm>

using namespace std;

string palindrome(string s)

{

string a=s;

reverse(a.begin(), a.end());

if(a==s){

return "Palindrome";

}

else{

return "Not Palindrome";

}

}

int main()

{

string s;

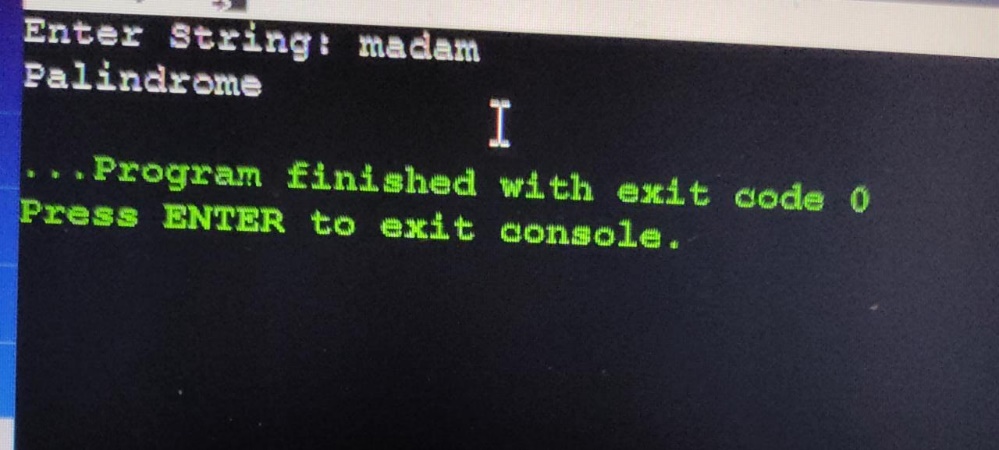
cout<<"Enter String: ";

cin>>s;

cout<<palindrome(s);

}

**Output --:**



**PROGRAM NO. 5**

Print a given matrix in spiral form.

Input: {{1, 2, 3, 4},

{5, 6, 7, 8},

{9, 10, 11, 12},

{13, 14, 15, 16 }}

Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Source Code :

#include <iostream>

using namespace std;

void spiral()

{

int x,y;

cout<<"Enter number of rows";

cin >>x;

cout<<"Enter number of columns";

cin>>y;

int arr[x][y];

for (int i = 0; i < x; ++i) {

for (int j = 0; j < y; ++j) {

cin >> arr[i][j];

}

}

cout << "\n";

int cr = 0,cc = 0,m=x,n=y;

while (cr < m && cc < n) {

for (int i = cc; i <n; i++) {

cout << arr[cr][i] << " ";

}

cr++;

for (int i = cr; i < m; i++) {

cout << arr[i][n-1] << " ";

}

n--;

if(cr<m){

for (int i = n-1; i >= cc; i--){

cout << arr[m-1][i] << " ";

}

}

m--;

if(cc<n){

for (int i = m-1; i >= cr; i--){

cout << arr[i][cc] << " ";

}

}

cc++;

}

}

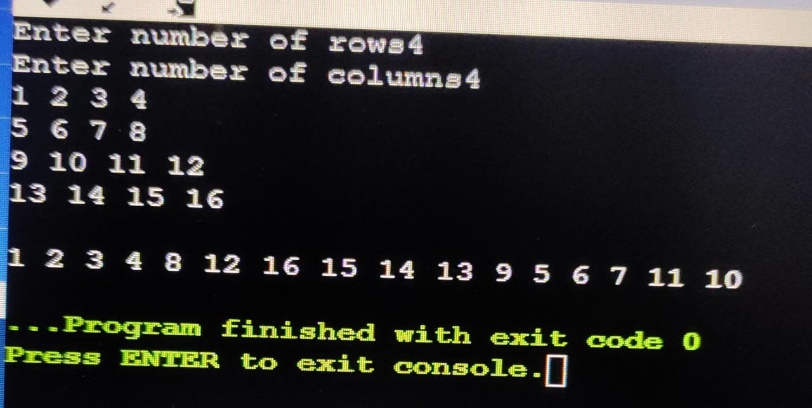
int main()

{

spiral();

}

**Output:**



**PROGRAM NO. 6**

An array of integers is given, both +ve and -ve. You need to find the two elements such

that their sum is closest to zero.

Example:

int arr[] = {1, 60, -10, 70, -80, 85};

For the above array, program should print -80 and 85.

Source Code:

#include <iostream>

#include<climits>

using namespace std;

void closetozero(int \*arr, int n)

{

int a,b,count = INT\_MAX;

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

for (int j = i+1; j < n; j++) {

int sum = arr[i] + arr[j];

if (sum >= 0 && sum < count){

count = sum;

a = i;

b = j; }

}

}

cout<<"Elements having sum nearest to zero are : ";

cout << arr[a] << " " << arr[b] << endl;

}

int main()

{

int x;

cout<<"Enter size of array : ";

cin >>x;

cout<<"Enter array : ";

int arr[x];

for (int i = 0; i < x; i++) {

cin >> arr[i];

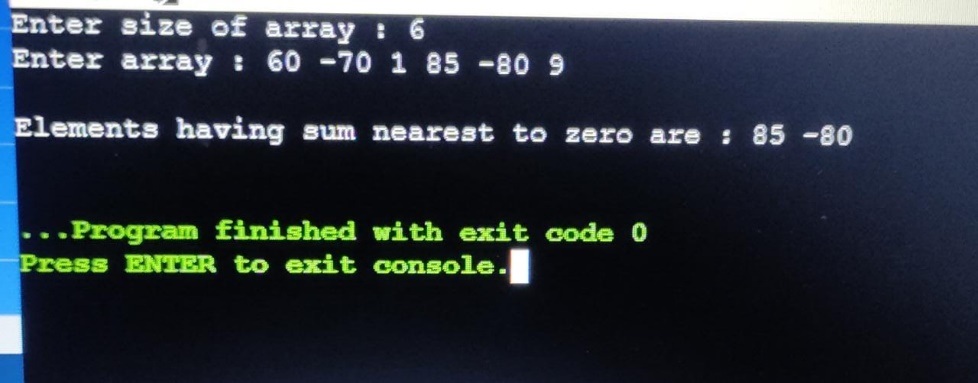
}

cout << "\n";

closetozero(arr,x);

}

**Output:**



**PROGRAM NO. 7**

Write a class for the following scenario and run it.

The employees have four attributes: id, name, salary, and company name. Initialize id,

name, and salary at the time of object creation. The list of methods is given as

void setID(int) - to set the employee’s ID.

void setName(string) - to set the employee’s name

void setSalary(int) - to set the employee’s salary.

void setCompany\_name(string)- to set the employee’s company name

int getId()-to get the employee’s id

string getName()-to get the employee’s name.

int getSalary()-to get the employee’s salary.

string getCompany\_name()-to get the employee’s company name.

Source Code:

#include<iostream>

using namespace std;

class emp{

private:

string name,company;

int empID,sal;

public:

void setdata(string n, int id, int s,string c){

name=n;

empID=id;

sal=s;

company=c;

}

void show() {

cout<<"Name Of Employee: "<<name<<endl;

cout<<"ID Of Employee: "<<empID<<endl;

cout<<"Salary Of Employee: "<<sal<<endl;

cout<<"Company of the Employee: "<<company<<endl;

}

};

int main()

{

string n,c;

int id,sal;

emp d;

cout<<"Enter Employee Name: ";

cin>>n;

cout<<"Enter Employee ID: ";

cin>>id;

cout<<"Enter Employee Salary: ";

cin>>sal;

cout<<"Enter Employee Company: ";

cin>>c;

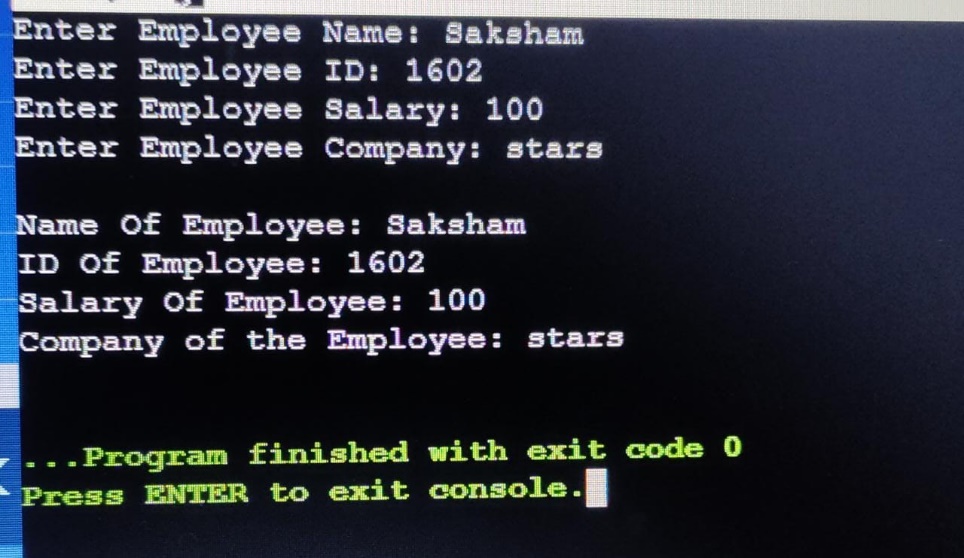
cout<<endl;

d.setdata(n,id,sal,c);

d.show();

}

**Output:**



**PROGRAM NO. 8**

Write a program to construct objects of a class with provided initial values. Also,

demonstrate the copy constructor.

Source Code:

#include<iostream>

using namespace std;

class emp

{

private:

string name;

int empID;

int sal;

public:

emp(){

}

emp(string n, int id, int s){

name=n;

empID=id;

sal=s;

}

emp(emp &d){

name=d.name;

empID=d.empID;

sal=d.sal;

}

void show(){

cout<<"Name of the Employee: "<<name<<endl;

cout<<"ID of Employee: "<<empID<<endl;

cout<<"Salary of Employee: "<<sal<<endl;

cout<<"\n";

}

};

int main()

{

string n;

int id,s;

cout<<"Enter Name of employee: ";

cin>>n;

cout<<"Enter ID of employee: ";

cin>>id;

cout<<"Enter Salary of employee: ";

cin>>s;

cout<<endl;

emp d1(n,id,s);

d1.show();

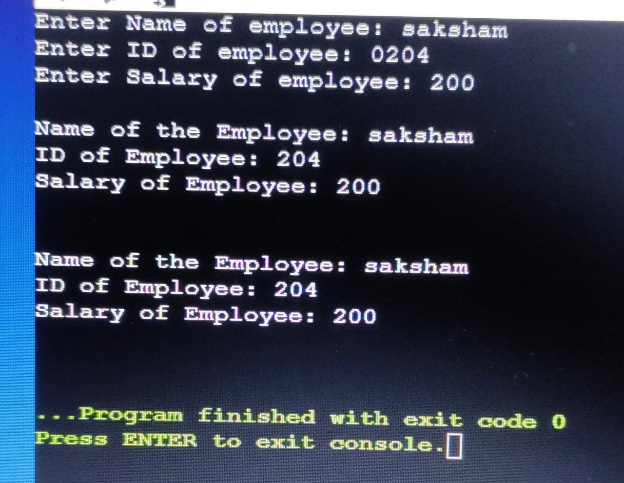
cout<<endl;

emp d2(n,id,s);

d2.show();

}

**Output:**



**PROGRAM NO. 9**

Write a program that counts the total number of objects created for any given class.

Source Code:

#include <iostream>

using namespace std;

static int count= 0;

class A

{

private:

int x;

public:

A()

{

count++;

x = 10;

}

};

int main(){

A d1;

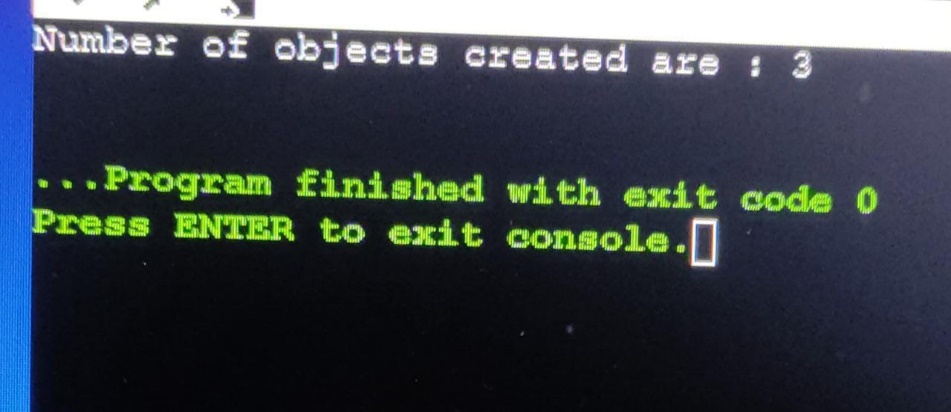
A d2;

A d3;

cout <<"Number of objects created are : "<<count<<endl;

}

**Output:**



**PROGRAM NO. 10**

If you want to access private members of a class, then what are the options? Use

examples to demonstrate.

Friend function is used to access the private member of the class.

Source Code:

#include<iostream>

using namespace std;

class A

{

private:

int x,y;

public:

void setdata(int x1, int y1) {

x=x1;

y=y1;

}

friend void sum(A &);

};

void sum(A &a)

{

cout<<"The sum is: "<<a.x+a.y;

}

int main()

{

A d;

int g,h;

cout<<"Enter First Value: ";

cin>>g;

cout<<"Enter Second Value: ";

cin>>h;

cout<<endl;

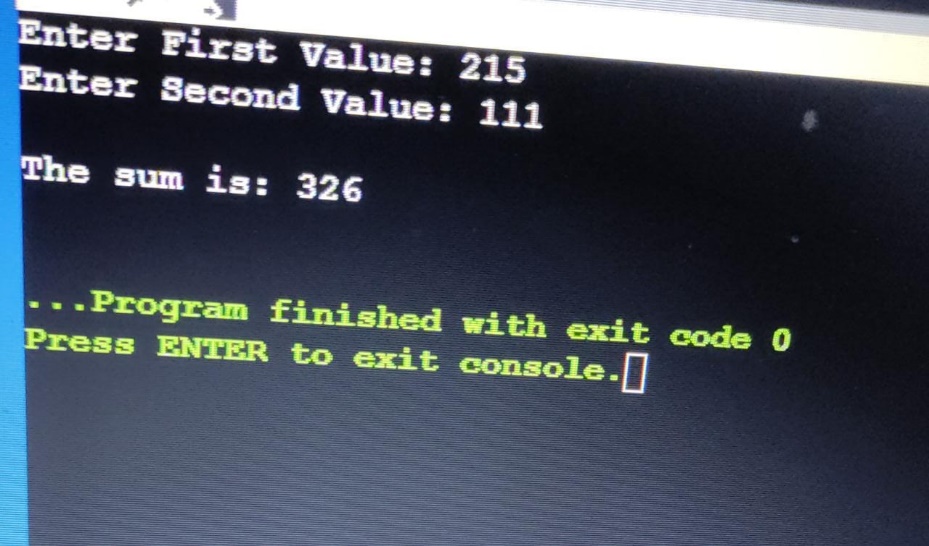
d.setdata(g,h);

sum(d);

cout<<endl;

}

**Output:**



**PROGRAM NO. 11**

Write a program to create objects at the time of run. Also, write code to delete objects

dynamically.

Source Code:

#include<iostream>

using namespace std;

class A

{

private:

int x,y;

public:

A(int a,int b) {

x=a;

y=b;

}

void show()

{

cout<<"Value of x: "<<x<<endl;

cout<<"Value of y: "<<y<<endl;

}

~A()

{

cout<<"The object is deleted";

}

};

int main()

{

int a,b;

cout<<"Enter first value: ";

cin>>a;

cout<<"Enter second value: ";

cin>>b;

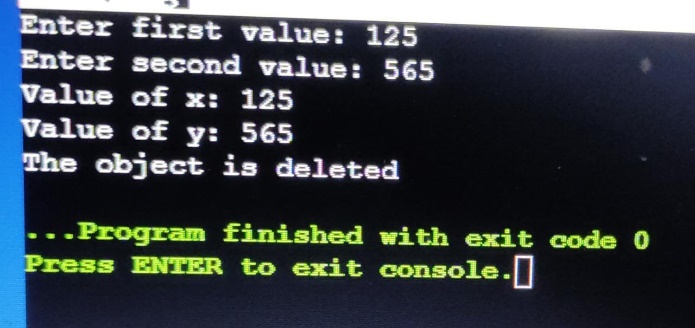
A \*p=new A(a,b);

p->show();

delete p;

}

**Output:**



**PROGRAM NO. 12**

Write a program to overload the ++ operator for the following scenario.

Class A

Attributes are as follows: int a=1, b=2, c=3.

Overload the ++ operator (for both prefix and postfix) so that when you apply it to an

object of class A, it increases the values of all attributes by one. For this example, the

answer will be

a=2, b=3 and c=4.

Source Code:

#include<iostream>

using namespace std;

class A

{

private:

int x,y,z;

public:

A(int a,int b,int c)

{

x=a;

y=b;

z=c;

}

void show() {

cout<<"x= "<<x<<endl;

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

cout<<"z= "<<z<<endl;

}

void operator ++(){ //for prefix

x++;

y++;

z++;

}

void operator ++(int){ //for postfix

x++;

y++;

z++;

}

};

int main()

{

int a,b,c;

cout<<"Enter values : "<<endl;

cin>>a>>b>>c;

A d(a,b,c);

cout<<"Prefix"<<endl;

++d;

cout<<"After updation: "<<endl;

d.show();

cout<<"Postfix"<<endl;

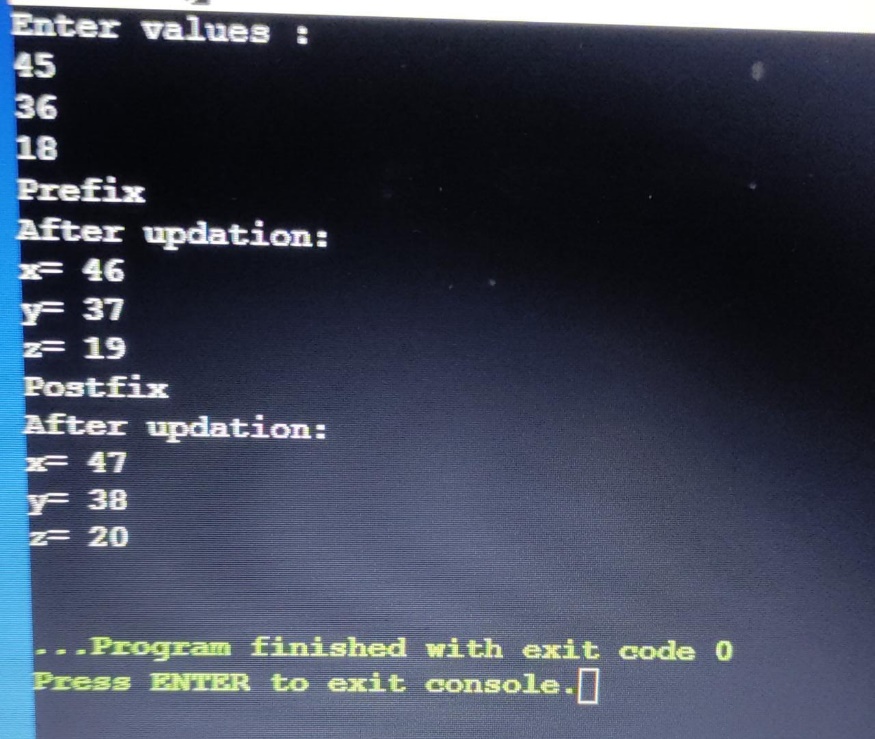
d++;

cout<<"After updation: "<<endl;

d.show();

}

**Output:**



**PROGRAM NO. -13**

Overload the binary operator + to add the integer attribute values of two objects.

Source code -

#include<iostream>

using namespace std;

class A

{

private:

int x,y,z;

public:

A(){

}

A(int a,int b,int c){

x=a;

y=b;

z=c;

}

void show()

{

cout<<"Sum is : ";

cout<<x<<" "<<y<<" "<<z;

}

A operator +(A a){

A g;

g.x=x+a.x;

g.y=y+a.y;

g.z=z+a.z;

return g;

}

};

int main()

{

int a,b,c;

cout<<"Enter Values : "<<endl;

cin>>a>>b>>c;

A d1(a,b,c);

cout<<"Renter values : "<<endl;

cin>>a>>b>>c;

A d2(a,b,c);

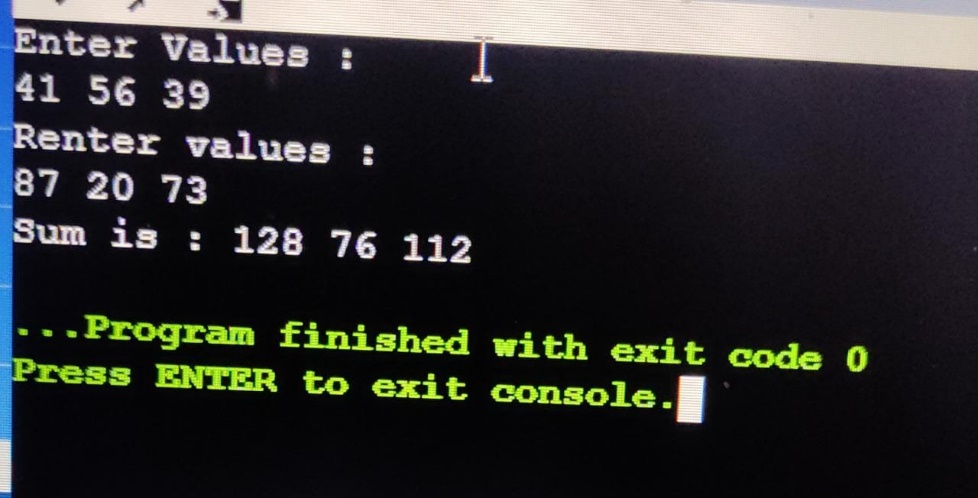
A d3;

d3=d1+d2;

d3.show();

}

**Output -:**



**PROGRAM NO. -14**

Overload the binary operator + to add two strings.

Source Code-

#include<iostream>

#include<string.h>

using namespace std;

class A

{

private:

string x;

public:

A(){

}

A(string a){

x=a;

}

void show()

{

cout<<"String is : ";

cout<<x;

}

A operator +(A a){

A g;

g.x=x+a.x;

return g;

}

};

int main()

{

string a;

cout<<"Enter Values : "<<endl;

cin>>a;

A d1(a);

cout<<"Renter values : "<<endl;

cin>>a;

A d2(a);

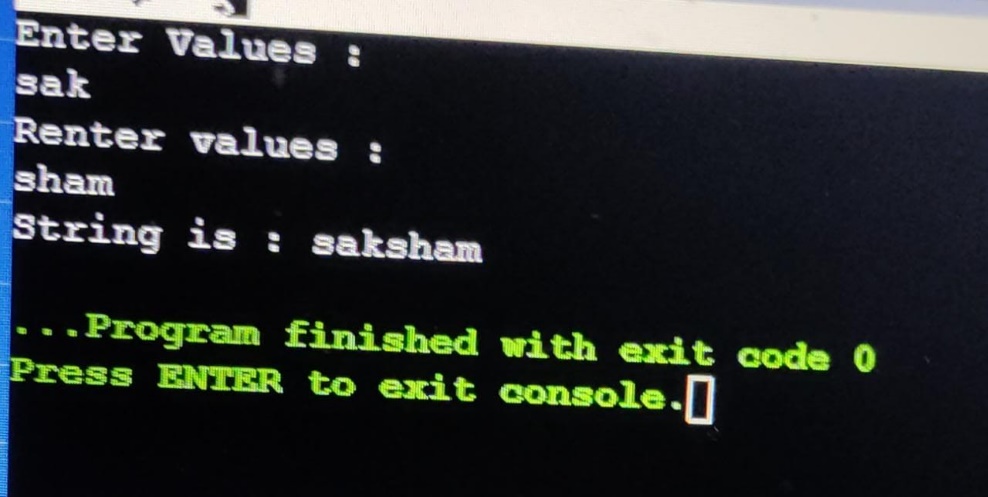
A d3;

d3=d1+d2;

d3.show();

}

**Output -:**



**PROGRAM NO. -15**

Create a base class with attributes name, age and roll\_number. Consider name as private, age as protected, and roll\_number as public. Write various child programs that inherit the base class in different access modes like private, protected and public and show the accessibility of various attributes.

Source Code-

#include<iostream>

using namespace std;

class A

{

private:

string name;

protected:

int age;

public:

int roll\_number;

void set\_details()

{

cout<<"Enter Details: "<<endl;

cin>>name;

cin>>age;

cin>>roll\_number;

}

void display()

{

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

cout<<"Age : "<<age<<endl;

cout<<"Roll Number : "<<roll\_number<<endl;

}

};

class B : private A

{

};

class C : protected A

{

};

class D : public A

{

};

int main()

{

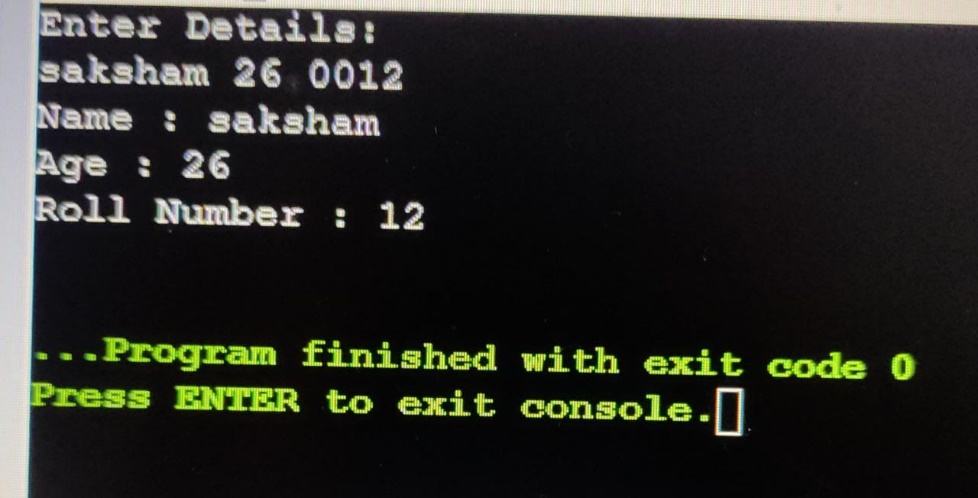
D d;

d.set\_details();

d.display();

}

**Output -:**



**PROGRAM NO. -16**

Write a base class having two attributes int x, int y. Write child class having void sum(int x,int y) function to add two provided values. You have to initialize base class’s attributes with the help of child class object and then use sum method to add the values. Also write a show() method in child to show the result. Hint: Use parameterized constructor.

Source Code-

#include<iostream>

#include<string.h>

using namespace std;

class A

{

public:

int x,y,z;

A(int a,int b)

{

x=a;

y=b;

}

void sum()

{

z=x+y;

}

};

class B:public A

{

public:

B(int a,int b):A(a,b)

{

}

void show()

{

cout<<"Sum is: "<<z;

}

};

int main()

{

int a,b;

cout<<"Enter values: "<<endl;

cin>>a>>b;

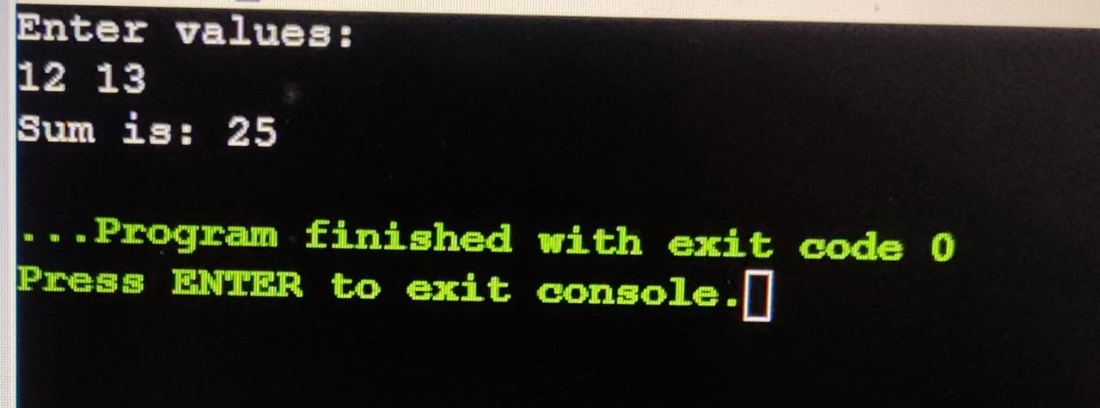
B d(a,b);

d.sum();

d.show();

}

**Output -:**



**PROGRAM NO. -17**

Write a program to copy the value of one object to another.

Source Code-

#include <iostream>

using namespace std;

class A

{

private:

int x, y;

public:

A(int x, int y)

{

this->x = x;

this->y = y;

}

A(const A &a)

{

x = a.x;

y = a.y;

}

int getX()

{

return x;

}

int getY()

{

return y;

}

};

int main()

{

int a,b;

cout<<"Enter values :"<<endl;

cin>>a>>b;

A d1(a,b);

A d2 = d1;

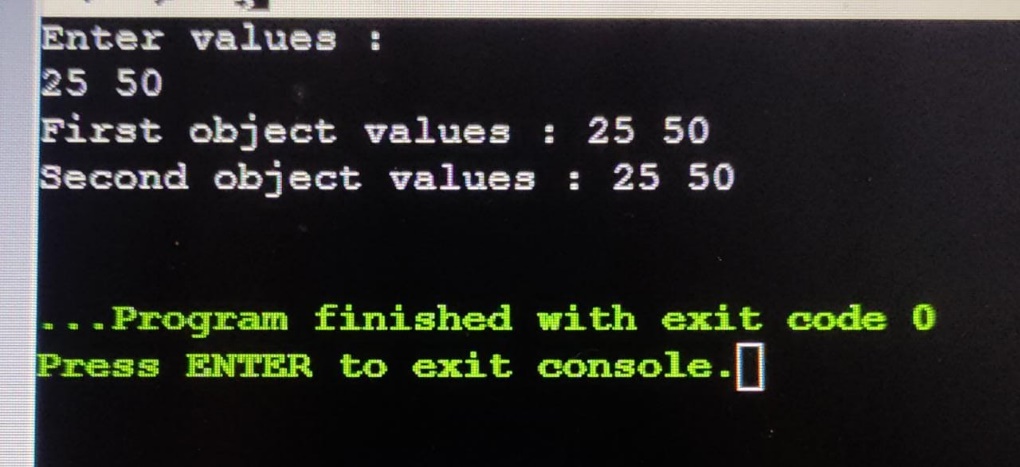
cout << "First object values : " << d1.getX()<<" "<<d1.getY()<<endl;

cout << "Second object values : "<< d2.getX()<<" "<<d2.getY()<<endl;

return 0;

}

**Output -:**



**PROGRAM NO. -18**

Write a program to demonstrate diamond’s problem. Also update the same program to

resolve it.

Source code-

Unresolved code:

#include <iostream>

using namespace std;

class A

{

public:

void display()

{

cout << "Hello"<<endl;

}

};

class B : public A {};

class C : public A {};

class D : public B, public C { };

int main()

{

D d;

d.display(); // Ambiguity

}

Resolved Code:

#include <iostream>

using namespace std;

class A

{

public:

void display()

{

cout << "Hello"<<endl;

}

};

class B : public virtual A {};

class C : public virtual A {};

class D : public B, public C { };

int main()

{

D d;

d.display();

}

**Output -:**



**PROGRAM NO. -19**

Write a program having a number of sum () methods which are used to sum the provided

arguments. Also write a show() method to show the result.

Source Code-

#include<iostream>

using namespace std;

class A

{

public:

int result;

void sum(int a)

{

result = a;

}

void sum(int a, int b)

{

result = a + b;

}

void sum(int a, int b, int c)

{

result = a + b + c;

}

void show()

{

cout<<"The sum is: "<<result<<endl;

}

};

int main()

{

int a,b,c;

cout<<"Enter Values : "<<endl;

cin>>a>>b>>c;

A a1, a2,a3;

a1.sum(a);

a1.show();

a2.sum(a,b);

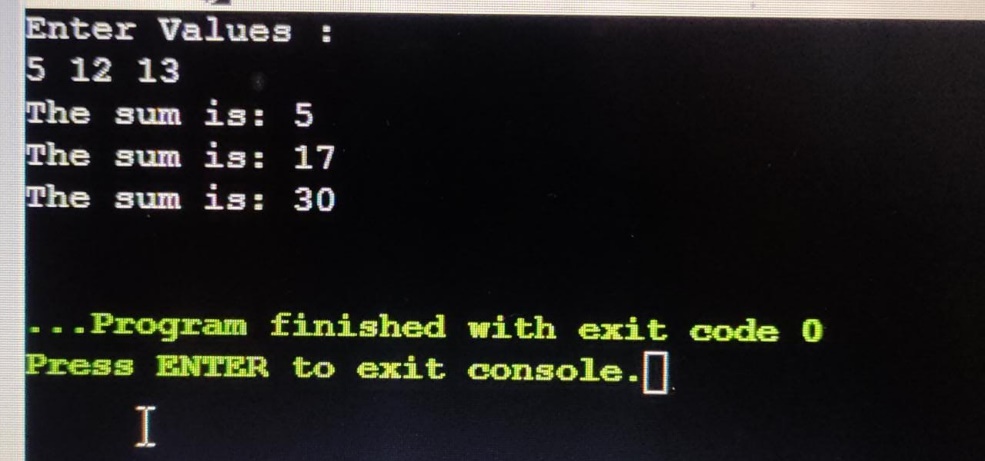
a2.show();

a3.sum(a,b,c);

a3.show();

}

**Output -:**



**PROGRAM NO. -20**

Write a program to demonstrate late binding.

Source Code-

#include<iostream>

using namespace std;

class Base

{

public:

int x=1;

virtual void show()

{

cout<<"Base Value : "<<x<<endl;

}

};

class Derived: public Base

{

int y =7;

void show()

{

cout<<"Derived Value : "<<y<<endl;

}

};

int main()

{

Base \*p;

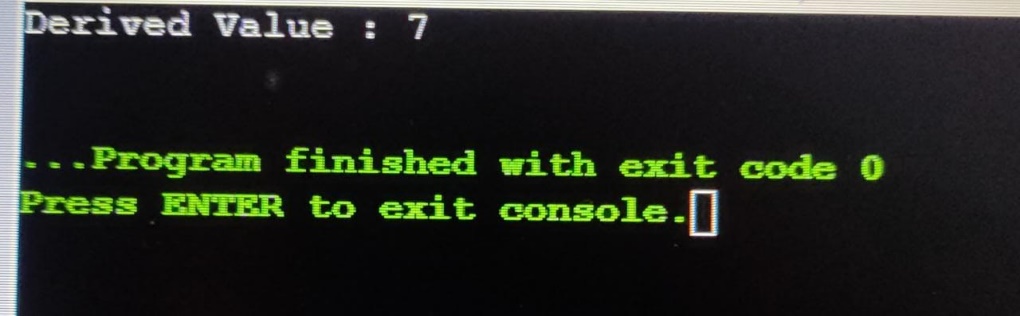
Derived d;

p=&d;

p->show();

}

**Output -:**



**PROGRAM NO. -21**

Write a program to demonstrate pure virtual functions and abstract classes.

Source Code-

#include<iostream>

using namespace std;

class Base

{

public:

int x;

virtual void display()=0;

};

class Derived: public Base

{

public:

int y;

Derived(int a): Base()

{

y=a;

}

void display()

{

cout<<"The value of second is : "<<y<<endl;

}

};

int main()

{

Derived d(100);

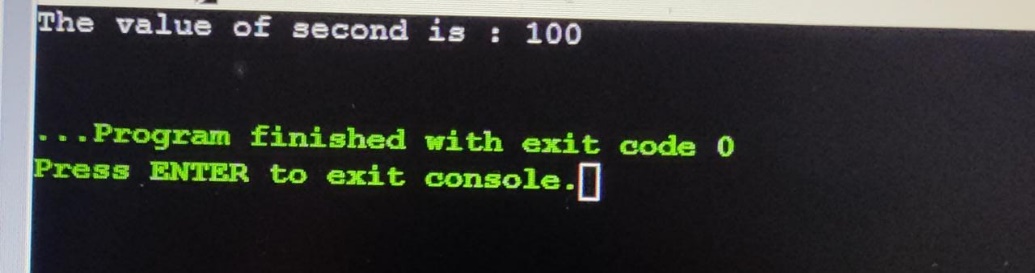
Base\* p[1];

p[0] = &d;

p[0]->display();

}

**Output -:**



**PROGRAM NO. -22**

If Parent and child classes are there and pointer of parent is referring to the object of child then how virtual destructor of parent plays an important role? Write a program to demonstrate the scenario.

Source Code-

#include<iostream>

using namespace std;

class Base

{

public:

Base()

{

cout<< "Constructor of Base class"<<endl;

}

virtual ~Base()

{

cout<< "Destructor of Base class"<<endl;

}

};

class Derived: public Base

{

public:

Derived()

{

cout << "Constructor of Derived class\n" ;

}

~Derived()

{

cout << "Destructor of Derived class\n" ;

}

};

int main()

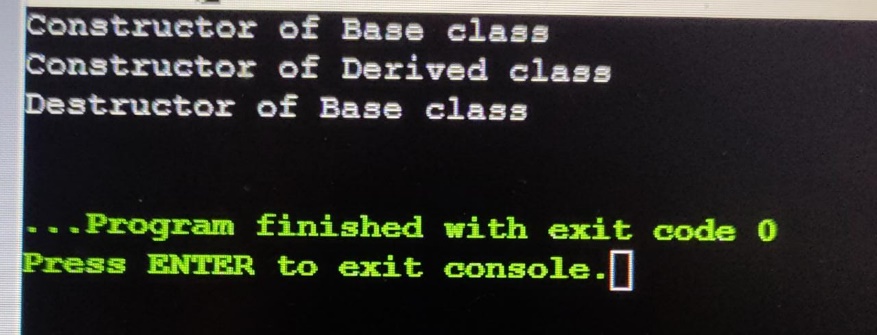
{

Base \*p = new Derived;

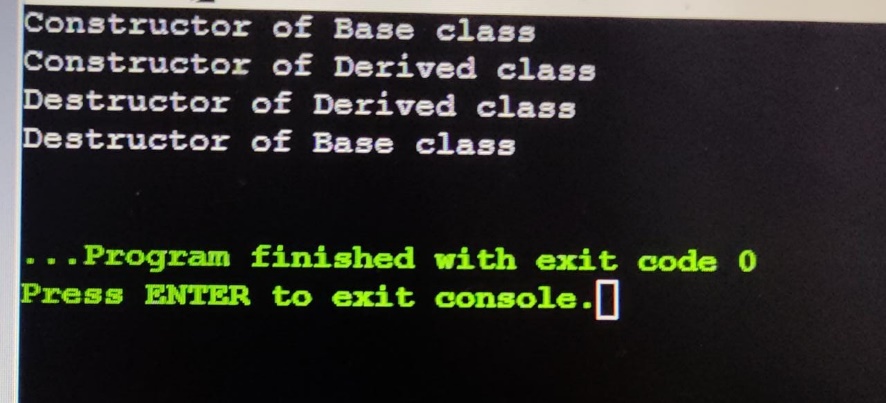
delete p;

}

Without using virtual destructor the output will be:



After using virtual destructor output is:



**PROGRAM NO. -23**

Write a program to read text from keyboard then write to some text file like “Sample.txt”. Then read this content and write back to console.

Source code-

#include<iostream>

#include<fstream>

using namespace std;

int main()

{

ofstream fout;

fout.open("abc.txt");

string a;

cin>>a;

fout<<a<<endl;

while(fout)

{

cin>>a;

if(a==".")

{

break;

}

fout<<a<<endl;

}

fout.close();

ifstream fin;

fin.open("abc.txt");

while(fin>>a)

{

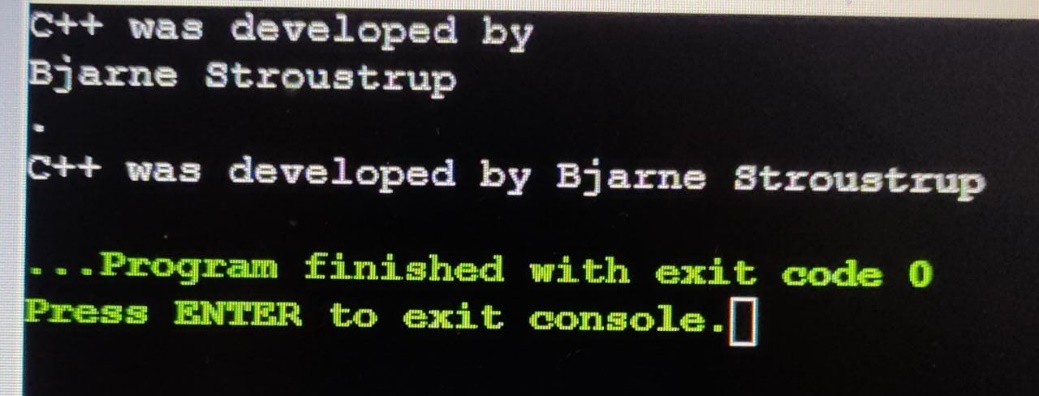
cout<<a<<" ";

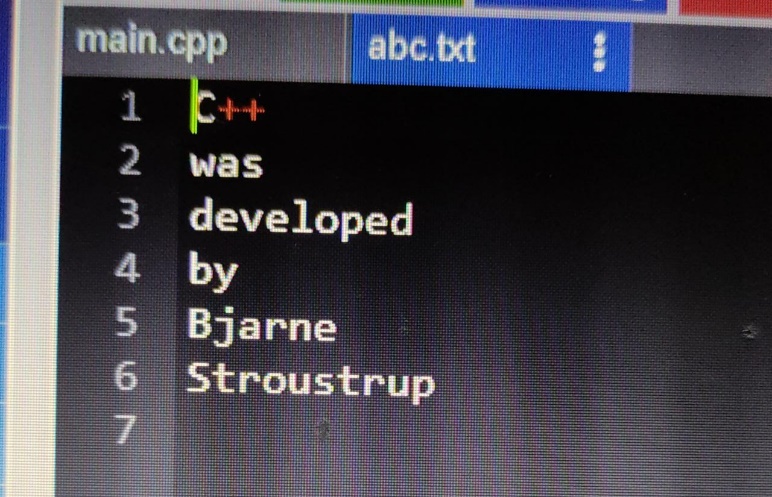
}

fin.close();

}

**Output -:**





**PROGRAM NO. -24**

Write a program to search any given word in a given file. Also show its frequency.

Source Code-

#include <iostream>

#include <fstream>

#include <string>

#include <algorithm>

using namespace std;

int main() {

cout << "Enter a word to search for: ";

string word;

cin >> word;

cout << "Enter the name of the file to search: ";

string file\_name;

cin >> file\_name;

ifstream file(file\_name);

if (!file.is\_open()) {

cout << "Error: Could not open file " << file\_name << endl;

return 1;

}

int count = 0;

string x;

while (file >> x) {

transform(x.begin(), x.end(), x.begin(), ::tolower);

if (x == word) {

count++;

}

}

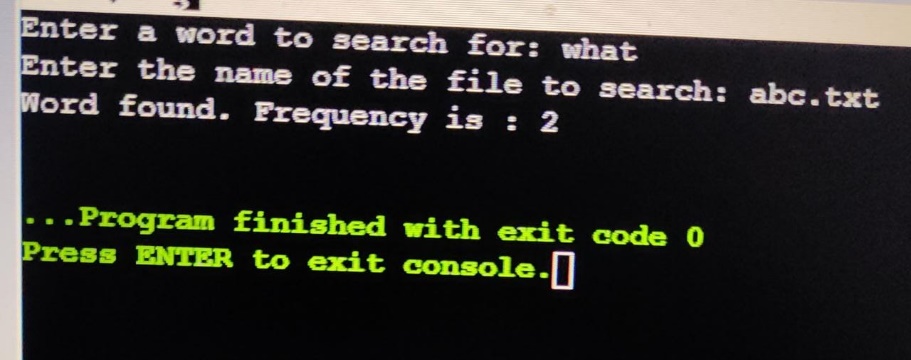
file.close();

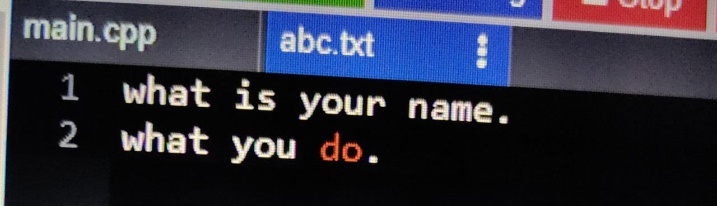
cout << "Word found. Frequency is : "<< count<< endl;

return 0;

}

**Output -:**





**PROGRAM NO. -25**

Write a program to open any existing file like sample.txt then write something in it and then read from start and display on console.

Source code-

#include<iostream>

#include <fstream>

#include <string>

using namespace std;

int main()

{

fstream fout;

fout.open("abc.txt");

if (!fout)

cout << "File not found."<<endl;

else {

fout<< "Welcome to programing world";

fout.close();

string word;

fout.open("abc.txt");

while (fout>> word) {

cout << word << " ";

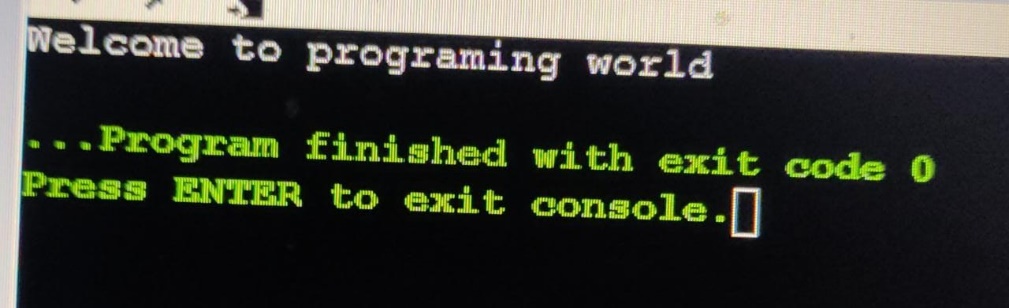
}

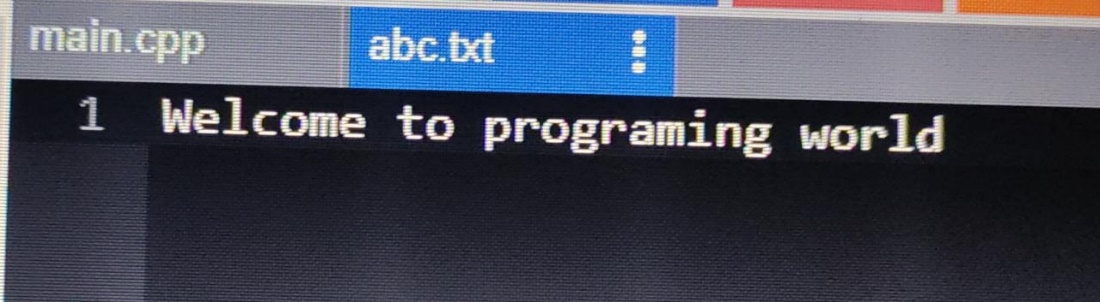
fout.close();

}

}

**Output -:**





**PROGRAM NO. -26**

Write a function sum(argument 1, argument 2) that can take any type of two arguments and add them. Write a program also to demonstrate.

Source Code-

#include <iostream>

using namespace std;

class A

{

public:

template <class T> T sum(T x, T y)

{

return x + y;

}

};

int main()

{

A d;

cout<<"Sum is : "<<endl;

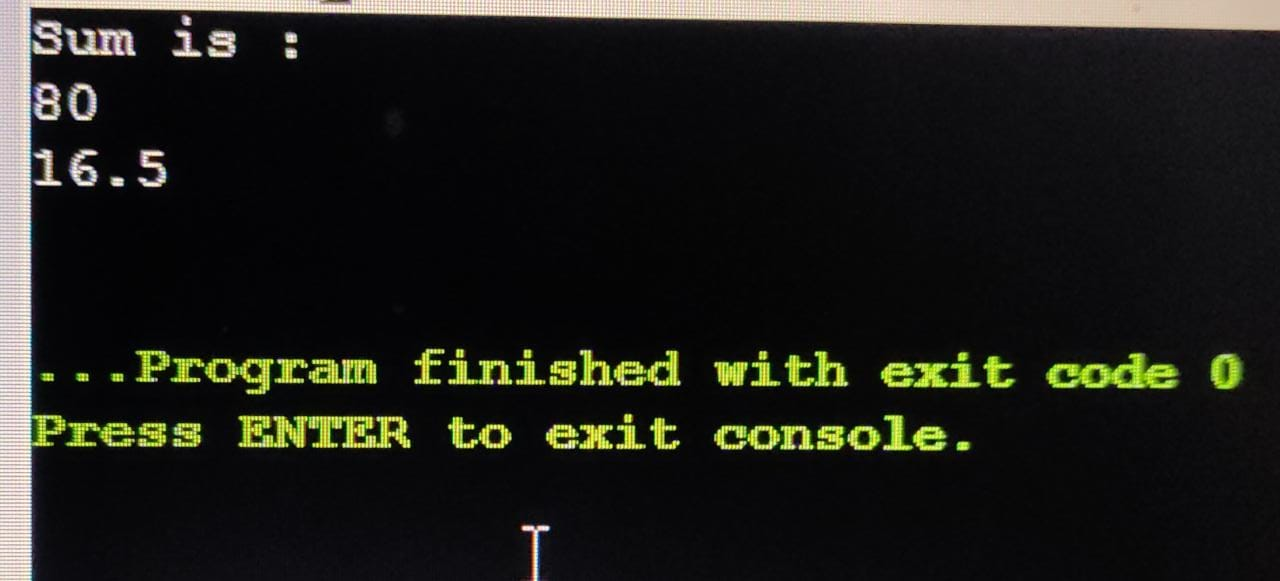
cout << d.sum(56,24) << endl;

cout << d.sum(6.9f,9.6f) << endl;

return 0;

}

**Output -:**



**PROGRAM NO. -27**

Overload generic function sum() as given in program no. 26 with three arguments.

Source Code-

#include <iostream>

using namespace std;

class A

{

public:

template <class T> T sum(T x, T y)

{

return x + y;

}

template <class T> T sum(T x, T y, T z)

{

return x+y+z;

}

};

int main()

{

A d;

cout<<"Sum is : "<<endl;

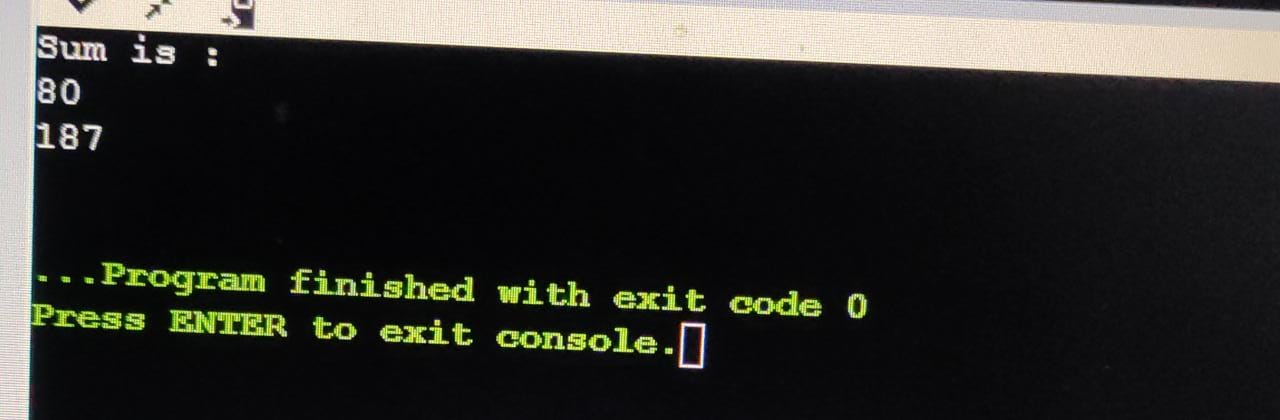
cout << d.sum(56,24) << endl;

cout << d.sum(75,30,82) << endl;

return 0;

}

**Output -:**



**PROGRAM NO. -28**

Write a program and handle exception that occurs during division and denominator is

zero.

Source Code-

#include<iostream>

using namespace std;

int main()

{

int x,y,z;

cout<<"Enter Numerator : "<<endl;

cin>>x;

cout<<"Enter Denominator : "<<endl;

cin>>y;

try

{

if(y==0){

throw 1;

}

else{

z=x/y;

cout<<"Answer is : "<<z<<endl;

}

}

catch(int a)

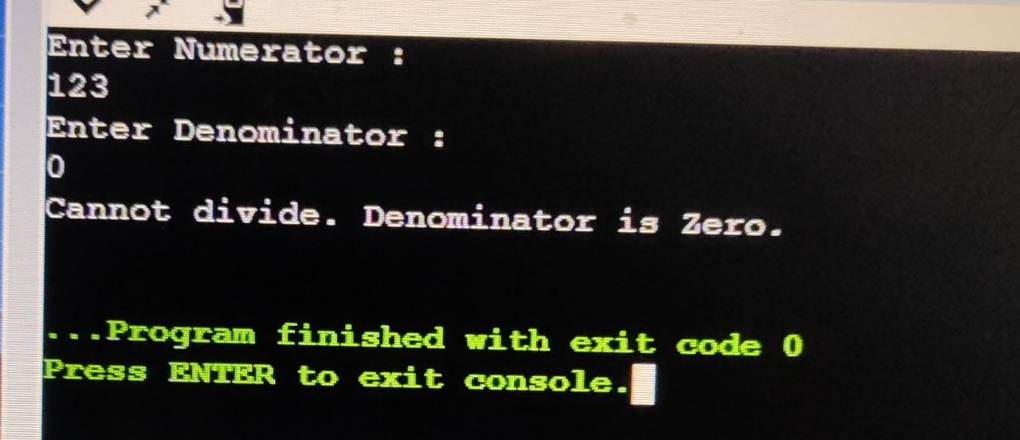
{

cout<<"Cannot divide. Denominator is Zero."<<endl;

}

}

**Output -:**



**PROGRAM NO. -29**

Create your own exception “MyException” that occurs at the same condition as mentioned in program no. 28.

Source Code-

#include<iostream>

#include<exception>

using namespace std;

class myexception:public exception

{

public:

const char \*what() const throw()

{

return "Cannot divide. Denominator is Zero.";

}

};

int main()

{

int x,y,z;

cout<<"Enter Numerator : "<<endl;

cin>>x;

cout<<"Enter Denominator : "<<endl;

cin>>y;

try

{

if(y==0){

myexception d;

throw d;

}

else{

z=x/y;

cout<<"Answer is : "<<z<<endl;

}

}

catch(myexception a)

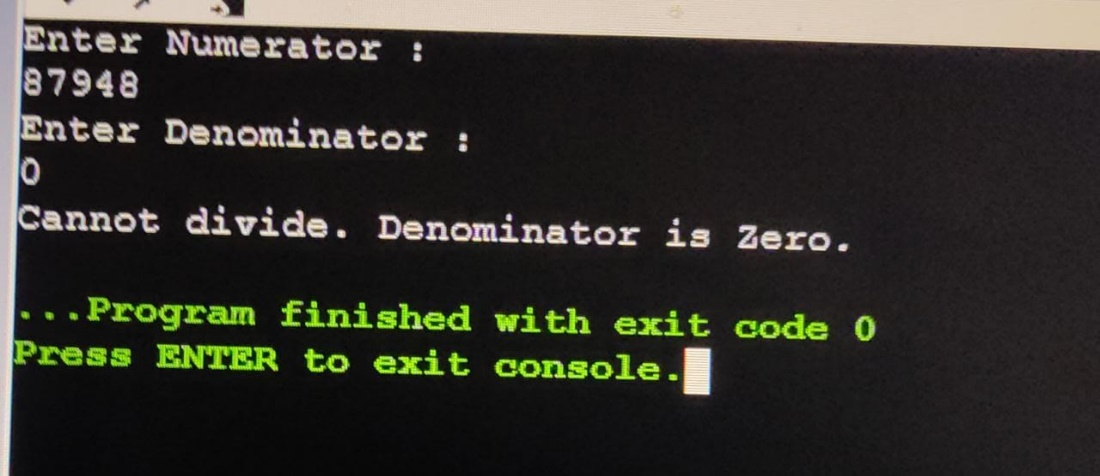
{

cout<<a.what();

}

}

**Output -:**



**PROGRAM NO. -30**

Create a vector and add five elements. Display all the elements, add a new element in last, show the capacity, remove from last, show the size, add element at start and 3rd position, remove all elements and then show the capacity again.

Source Code-

#include <iostream>

#include <vector>

using namespace std;

void display(vector<int>&V)

{

for(int i=0; i<V.size(); i++)

{

cout<<V[i]<<" ";

}

cout<<endl;

}

int main()

{

vector <int> V ;

cout<<"Enter Elements : "<<endl;

int z,a,b,c;

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

{

cin>>z;

V.push\_back(z);

}

display(V);

cout<<"Enter New Element to add : ";

cin>>a;

V.push\_back(a);

cout<<"Capacity is : "<<V.capacity()<<endl;

cout<<"Removing the last element"<<endl;

V.pop\_back();

cout<<"Size of Vector is : "<<V.size()<<endl;

cout<<"Enter element to add at start"<<endl;

cin>>b;

V.insert(V.begin(),b);

display(V);

cout<<"Enter element to add at 3rd positiont"<<endl;

cin>>c;

V.insert(V.begin()+2,c);

display(V);

V.clear();

cout<<"After Clearing"<<endl;

cout<<"Capacity: ";

cout<<V.capacity()<<endl;

cout<<"Size: ";

cout<<V.size()<<endl;

}

**Output -:**

