KARNATAK LAW SOCIETY’S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

**(APPROVED BY AICTE, NEW DELHI)**

Department of Computer Science Engineering



Name: Shradha Mallikarjun Patil

USN: 2GI20CS144

Semester: 2

Division: I

Roll Number: 2

Academic Year: 2020-2021

Course Name: Object Oriented Programming Using C++ Course

Code:18ACS28

Class: CS3

Faculty In-charge: Professor Prashant Nira

**INDEX**

|  |  |
| --- | --- |
| 1 | Program on classes and objects. |
| 2 | Program illustrating use of reference type in C++. |
| 3 | Program on function overloading. |
| 4 | Program on dynamic memory management in C++. |
| 5 | Program on array of objects. |
| 6 | Program on constructors and destructors. |
| 7 | Program on operator overloading. |
| 8 | Program on overloading i/o operators. |
| 9 | Program on inheritance. |
| 10 | Program on virtual function and pure virtual functions. |
| 11 | Program on dynamic polymorphism. |
| 12 | Program on file streams. |

**TERM WORK 1:**

**Program on classes and objects.**

Problem Statement:

1.1) Develop a C++ program consisting of a class called Employee with name, ID, department and basic salary as data members. Include member functions to: a. set values for data members b. compute gross salary where DA=70% of basic, HRA=20% of basic and deductions=5% of basic and c. Display details of an employee

CODE:

#include<iostream> using namespace std; class Employee

{

char name[100],dept[100]; int id;

float salary,gross; public:

void setdata(); void compute(); void display();

};

void Employee::setdata()

{

cout<<"Enter the name of employee="; cin>>name;

cout<<"Enter the employee id="; cin>>id;

cout<<"Enter the employee department="; cin>>dept;

cout<<"Enter the employee basic salary="; cin>>salary;

}

void Employee::compute()

{

float DA,HRA,deduction; DA=0.7\*salary;

HRA=0.2\*salary; deduction=0.05\*salary; gross=salary+DA+HRA-deduction;

}

void Employee::display()

{

cout<<"\nThe employee details are"<<endl; cout<<"The name of employee="<<name<<endl; cout<<"The employee id="<<id<<endl; cout<<"The employee department="<<dept<<endl;

cout<<"The basic salary of employee="<<salary<<endl; cout<<"The gross salary of employee="<<gross<<endl;

}

int main()

{

Employee e; system("cls");

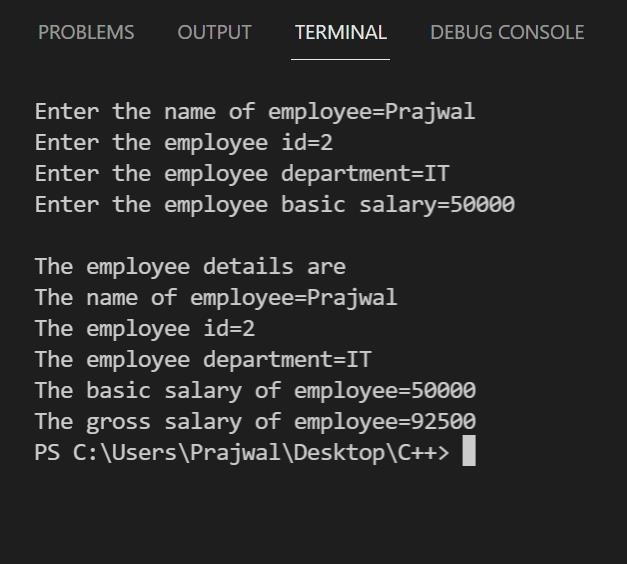
e.setdata();

e.compute();

e.display();

}

OUTPUT:



**TERM WORK 2:**

**Program illustrating use of reference type in C++**

Problem Statement:

Write a C++ program to swap two variables using reference variables.

CODE:

#include<iostream> using namespace std; void swap(int &x, int &y)

{

int temp; temp = x; x = y;

y = temp;

}

int main()

{

int a, b;

cout<<"Enter the value of a: "; cin>>a;

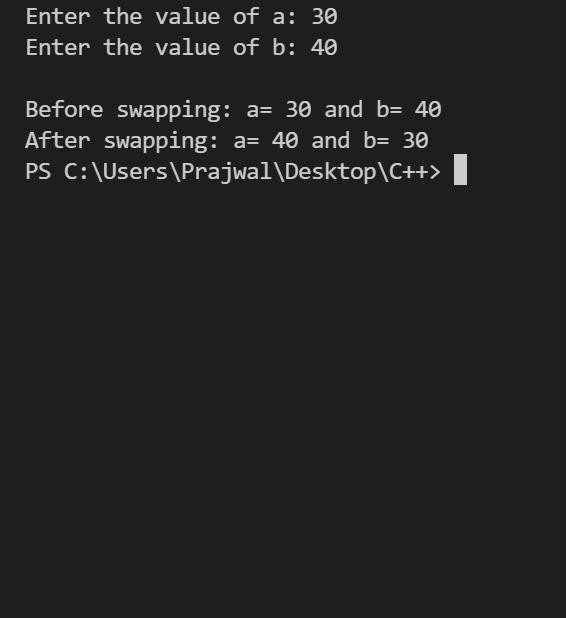
cout<<"Enter the value of b: "; cin>>b;

cout<<endl<<"Before swapping: "; cout<<"a= "<<a<<" and b= "<<b; swap(a, b);

cout<<endl<<"After swapping: "; cout<<"a= "<<a<<" and b= "<<b;

}

OUTPUT:



**TERM WORK:3**

**Program on function overloading**

Problem Statement

Write a C++ program to calculate the area of circle (given the radius), rectangle (given the two sides) and triangle (given the three sides) using function overloading.

CODE:

#include<iostream> #include<cmath> using namespace std; float area(float);

float area(float,float);

float area(float,float,float); int main()

{

float r,a,b,c,x,y;

system("cls");

cout<<"Enter the radius of circle="; cin>>r;

cout<<"Enter the length and breadth of rectangle="; cin>>x>>y;

cout<<"Enter the length of three sides of triangle="; cin>>a>>b>>c;

cout<<"\nArea of circle="<<area(r)<<endl; cout<<"\nArea of rectangle="<<area(x,y)<<endl; cout<<"\nArea of triangle="<<area(a,b,c)<<endl;

}

float area(float r)

{

float a; a=3.142\*r\*r; return (a);

}

float area(float x,float y)

{

float a; a=x\*y; return (a);

}

float area(float a,float b,float c)

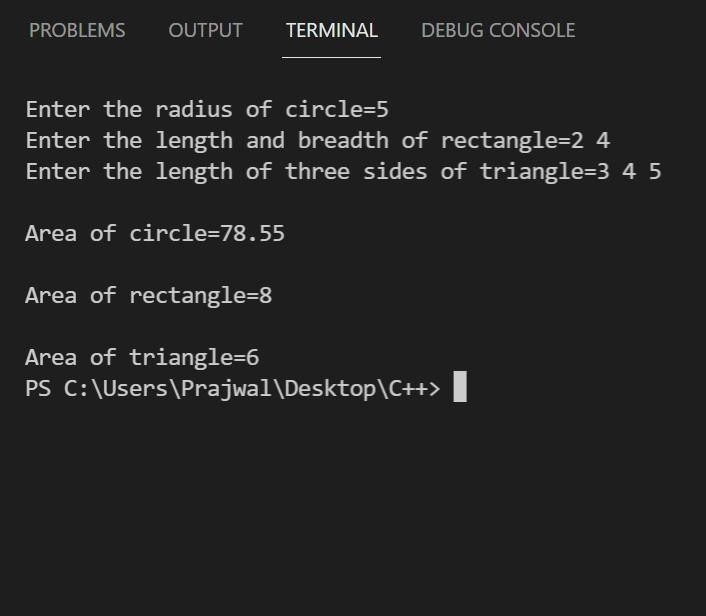
{

float s,area; s=(a+b+c)/2;

area=sqrt(s\*(s-a)\*(s-b)\*(s-c)); return (area);

}

OUTPUT:



TERM WORK:4

**Program on dynamic memory management in C++**

Problem Statement:

Create an array using dynamic memory allocation. Write functions to perform the following:

* + 1. Find the minimum element in the array
    2. Find the maximum element in the array
    3. Find the mean of the elements in the array

CODE:

#include <iostream> using namespace std;

void readArray(int a[], int n)

{

cout<< "Enter " << n << " elements:"; for(int i=0; i<n; i++)

cin>> a[i];

}

void findMin(int a[ ], int n)

{

int minimum=a[0]; for(int i=1; i<n; i++)

if(a[i]<minimum)

minimum = a[i];

cout<< "Minimum element is " << minimum <<endl;

}

void findMax(int a[], int n) { int maximum=a[0]; for(int i=1; i<n; i++)

if(a[i]>maximum)

maximum = a[i];

cout<< "Maximum element is " << maximum <<endl;

}

void findMean(int a[], int n) { int sum = 0;

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

sum += a[i];

cout<< "Mean of the elements is " << sum/n <<endl;

}

int main() {

int n;

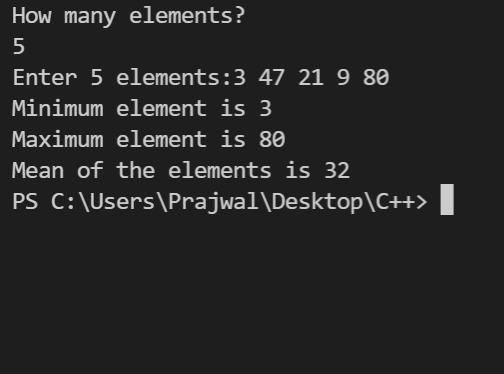
cout<< "How many elements?"<<endl; cin>> n;

int \*a = new int[n]; readArray(a, n); findMin(a, n);

findMax(a, n); findMean(a, n); return 0;

}

OUTPUT:



TERM WORK:5

**Program on array of objects**

Problem Statement:

Create a Book class containing data members viz., Book number, Title, Author and price. Write the main function that does the following:

* + 1. Creates an array of book objects and reads the book information using member function.
    2. Given the book number, searches and prints using friend function, the book details if found, an error message otherwise.

CODE:

#include<iostream> using namespace std; class Book

{

int number; char title[100];

char author[100]; float price; public:

void setdata(); void display();

friend void search(int,Book [ ]);

};

void Book::setdata()

{

cout<<"Enter the book number="; cin>>number;

cout<<"\nEnter the title of book="; fflush(stdin);

gets(title);

cout<<"\nEnter the name of author of book="; fflush(stdin);

gets(author);

cout<<"\nEnter the price of the book="; cin>>price;

}

void Book::display()

{

cout<<"\nthe book number="<<number; cout<<"\nthe title of book="<<title;

cout<<"\nthe name of author of book="<<author; cout<<"\nthe price of the book="<<price;

}

void search(int n,Book b[])

{

int i,no,flag=0;

cout<<"\nEnter the book number to search="; cin>>no;

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

{

if(b[i].number==no)

{

cout<<"\nBook found\n"; cout<<"Book details are\n"; b[i].display();

flag=1;

}

}

if(flag==0)

{

cout<<"\nBook not found";

}

}

int main()

{

Book b[10]; system("cls");

int n,i;

cout<<"Enter the total number of books="; cin>>n;

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

{

b[i].setdata();

}

cout<<"\nThe avaliable books are\n"; for(i=0;i<n;i++)

{

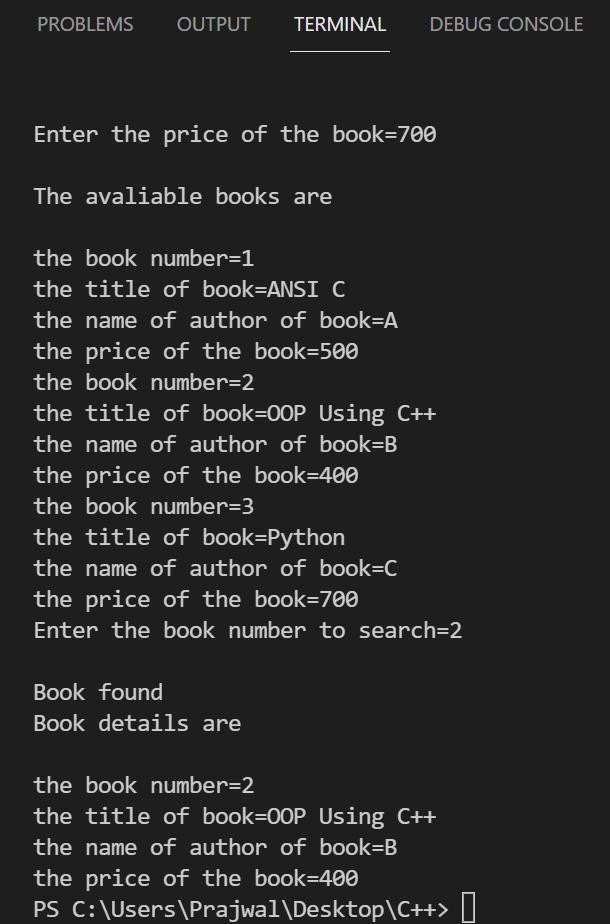
b[i].display();

}

search(n,b);

}

OUTPUT:



TERM WORK:6

**Program on constructors and destructors**

Problem Statement:

Create a class called intArray with Data members:

* + 1. pointer to an integer array and
    2. integer to hold the array length Member functions:

1. A zero-arg constructor
2. A parameterized constructor with an array and its length as parameters
3. Copy constructor
4. Display array elements
5. Destructor

Write the corresponding main()

CODE:

#include<iostream> using namespace std; class IntArray

{

private:

int len, \*arr;

public:

}

IntArray() {

len = 10;

arr = new int[len]; for(int i=0; i<len; i++)

arr[i]=0;

IntArray(int a[], int n) { len = n;

arr = new int[len]; for(int i=0; i<len; i++)

arr[i] = a[i];

}

IntArray(const IntArray&a)

{

len = a.len;

arr = new int[len];

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

arr[i] = a.arr[i];

}

void printArray()

{

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

cout<<setw(3) <<arr[i];

}

};

int main() {

cout<< "Creating a default IntArray object..." <<endl; IntArray ob1;

cout<< "Default IntArray object contents are:" <<endl; ob1.printArray();

cout<<endl<< "Creating parameterized IntArray object..." <<endl; int a[] = {1,2,3,4,5};

IntArray ob2(a,5);

cout<< "Parameterized IntArray object contents are:" <<endl; ob2.printArray();

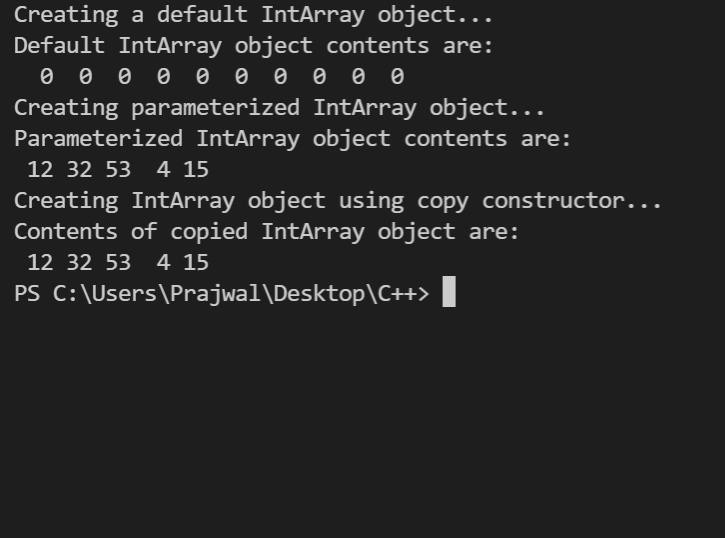
cout<<endl<< "Creating IntArray object using copy constructor..."<<endl; IntArray ob3(ob2);

cout<< "Contents of copied IntArray object are:" <<endl; ob3.printArray();

return 0;

}

OUTPUT:



TERM WORK:7

**Program on operator overloading**

Problem Statement:

Create a class called Complex with real and imaginary parts as data members. Member functions:

* + 1. A zero-argument constructor
    2. A parameterized constructor
    3. Overloaded + operator to add two complex numbers and return the sum
    4. Display complex number Write the corresponding main()

CODE:

#include <iostream> using namespace std; class Complex { private:

int real;

int imaginary; public:

Complex() {

real = imaginary = 0;

}

Complex(int r, int i) { real = r; imaginary = i;

}

void print() {

if(imaginary>0)

cout<< real << "+i" << imaginary <<endl;

else

}

cout<<real<<”-i”<<(-1)\*imaginary<<endl;

friend Complex operator+(Complex a, Complex);

};

Complex operator+(Complex a, Complex b) { Complex res;

res.real = a.real + b.real;

res.imaginary = a.imaginary + b.imaginary; return res;

}

int main() {

Complex c1(6,2), c2(2,5); cout<< "Complex c1 = "; c1.print();

cout<< "Complex c2 = "; c2.print();

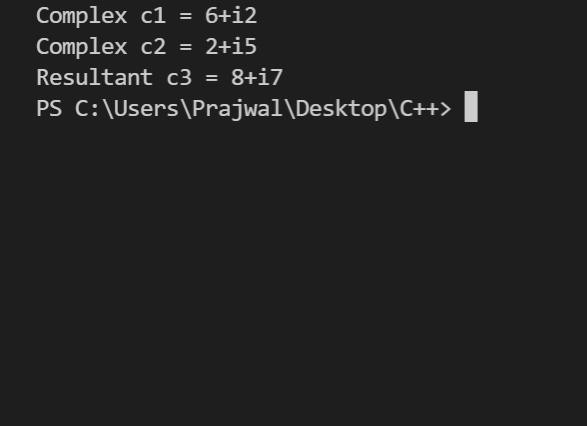
Complex c3; c3 = c1 + c2;

cout<<"Resultant c3 = "; c3.print();

return 0;

}

OUTPUT:



TERM WORK:8

**Program on overloading i/o operators**

Problem Statement:

Create a class called Distance with feet and inches as data members. Member functions:

* + 1. A zero-argument constructor
    2. A parameterized constructor Friend functions:

1. Overloaded extraction operator to read a distance
2. Overloaded insertion operator to display a distance (Ex - 10 feet 5 inches)
3. Compare two distances Write the corresponding main()

CODE:

#include<iostream> using namespace std; class distanc

{

float feet,inch; public:

distanc(){ feet=inch=0;} distanc(float,float);

friend istream & operator >>(istream &,distanc &); friend ostream & operator <<(ostream &,distanc &); friend int comp(distanc,distanc);

};

distanc::distanc(float f,float i)

{

feet=f; inch=i;

}

istream & operator >>(istream & in,distanc &d)

{

cout<<"Enter feet="<<endl; in>>d.feet;

cout<<"Enter inch="<<endl; in>>d.inch;

return in;

}

ostream & operator <<(ostream &out,distanc &d)

{

cout<<"Distance is="<<endl;

out<<d.feet<<"feet"<<d.inch<<"inch"<<endl; return out;

}

int comp(distanc d1,distanc d2)

{

if((d1.feet==d2.feet)&&(d1.inch==d2.inch))

{

return 0;

}

else if((d1.feet==d2.feet)&&(d1.inch>d2.inch))

{

return 1;

}

else if(d1.feet>d2.feet)

{

return 1;

}

else

{

return -1;

}

}

int main()

{

distanc d1,d2; system("cls"); cin>>d1; cout<<d1; cin>>d2;

cout<<d2; if(comp(d1,d2)==0)

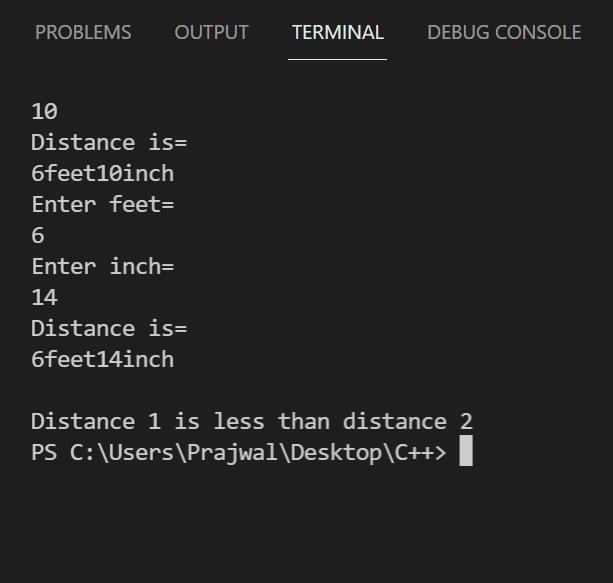
cout<<"\ndistances are equal"; if(comp(d1,d2)==1)

cout<<"\ndistance 1 is greater than distance 2"; if(comp(d1,d2)==-1)

cout<<"\nDistance 1 is less than distance 2";

}

OUTPUT:



TERM WORK:9

**Program on inheritance**

Problem Statement:

Create a class called Manager with attributes: name, ID and basic salary. Demonstrate inheritance by deriving classes HR Manager and Sales Manager from Manager and compute gross salary as per the following:

HR Manager - DA = 70% of basic, HRA = 20% of basic, deductions = 5% of basic Sales Manager - DA = 70% of basic, HRA = 10% of basic, TA = 5% of basic, deductions = 5% of basic

CODE:

#include<iostream> using namespace std; class Manager

{

protected: string name; int id;

float basic; public:

Manager(string n, int i, float b)

{

name=n; id=i; basic=b;

}

void disp()

{

cout<<"\n Manager details"<<endl; cout<<name<<"\t"<<id<<"\t"<<basic<<endl;

}

};

class HRmanager: public Manager

{

float gross,da,hra,ded; public:

HRmanager(string n,int i,float b):Manager(n,i,b){} float computeGross()

{

da=0.7 \*basic; hra=0.2\*basic; ded=0.05\*basic; gross=(basic+da+hra)-ded; return gross;

}

void disp()

{

Manager::disp();

cout<<"Gross salary="<<computeGross()<<endl;

}

};

class Salesmanager: public Manager

{

float gross,da,hra,ded,ta; public:

Salesmanager(string n,int i,float b):Manager(n,i,b){} float computeGross()

{

da=0.7 \*basic; hra=0.1\*basic; ta=0.05\*basic; ded=0.05\*basic;

gross=(basic+da+hra+ta)-ded; return gross;

}

void disp()

{

Manager::disp();

cout<<"Gross salary="<<computeGross()<<endl;

}

};

int main()

{

system("cls");

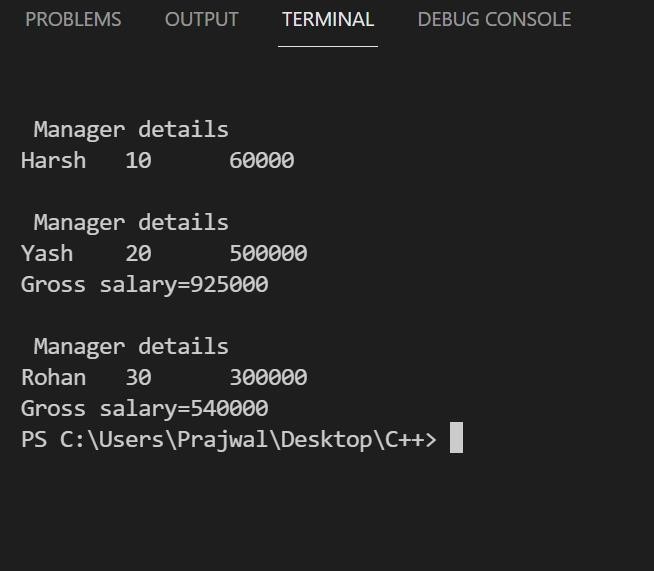
Manager m("Harsh",10,50000); m.disp();

HRmanager x("Rajat",20,600000); x.disp();

Salesmanager s("Rohan",30,400000); s.disp();

}

OUTPUT:



TERM WORK:10

**Program on virtual function and pure virtual functions**

Problem Statement:

Create a base class called Student and derived classes UGStudent and PGStudent. Demonstrate polymorphism using virtual functions to compute the total score as follows:

UGStudent - average of best two scores PGStudent - addition of two scores

CODE:

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

class Student { protected:

int m1, m2, m3, avg;

public:

Student(int m1, int m2, int m3) { this->m1 = m1;

this->m2 = m2; this->m3 = m3;

}

Student(int m1, int m2) { this->m1 = m1; this->m2 = m2;

}

virtual void findTotal() { }

};

class UGStudent : public Student { public:

UGStudent(int m1, int m2, int m3) : Student(m1, m2, m3) {} void findTotal() {

int smallest = (m1<m2) ? (m1<m3 ? m1 : m3) : (m2<m3 ? m2 : m3); avg = ceil((m1+m2+m3-smallest)/2.0);

cout << "Average score of UG Student is " << avg << endl;

}

};

class PGStudent : public Student { public:

PGStudent(int m1, int m2) : Student(m1, m2) {} void findTotal() {

avg = m1+m2;

cout << "Total score of PG Student is " << avg << endl;

}

};

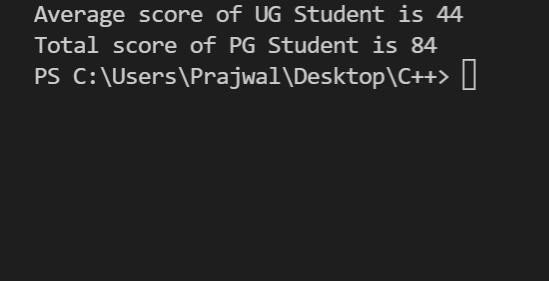
int main() {

UGStudent ug(37,48,47); PGStudent pg(41,43); ug.findTotal(); pg.findTotal();

return 0;

}

OUTPUT:



TERM WORK:11

**Program on dynamic polymorphism**

Problem Statement:

Create a base class called List that has virtual functions: CreateList and DispList. Create two classes IntList and CharList that inherit from List. Both classes override CreateList and DispList. The IntList class has an additional member function to return the sum of elements of the list of integers. The CharList has an additional member function to search for a character received as a parameter and returns the position of it’s first occurrence, if found in the list of characters; else returns -1. Demonstrate dynamic polymorphism by creating and upcasting objects of the two derived types in main

CODE:

#include<iostream> #include<cstring> using namespace std; class List {

public:

virtual void createList() = 0; virtual void dispList() = 0;

};

class IntList : public List { private:

int \*p; int n;

public:

void createList() {

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

p = new int[n];

cout << "Enter " << n << " elements:" << endl; for(int i=0; i<n; i++)

cin >> p[i];

}

void dispList() {

cout << "The list has following elements:" << endl; for(int i=0; i<n; i++)

cout << p[i] << " "; cout << endl;

}

int findSum() {

int sum = 0;

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

sum += p[i]; return sum;

}

};

class CharList : public List { private:

char \*p; int n;

public:

void createList() {

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

cin.ignore(80,'\n'); p = new char[n+1];

cout << "Enter " << n << " characters:" << endl; int i;

char ch;

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

cin.get(p[i]);

cin.ignore(80,'\n');

}

p[i] = '\0';

}

void dispList() {

cout << "The character list is " << p << endl;

}

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

if(p[i] == ch)

return i;

return -1;

}

};

int main() { system("cls");

IntList arr1; List \*p = &arr1; p->createList(); p->dispList();

cout << "Sum of integet list is " << arr1.findSum() << endl; CharList arr2;

p = &arr2;

p->createList(); p->dispList(); char ch;

cout << "Enter character to search:"; cin >> ch;

int position = arr2.searchChar(ch); if(position == -1)

cout << ch << " is not found in the list" << endl;

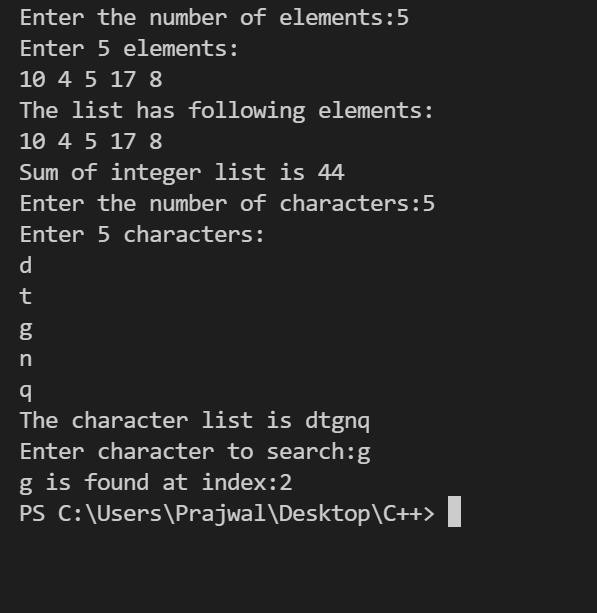
else

cout << ch << " is found at index" << position << endl;

return 0;

}

OUTPUT:



TERM WORK:12

**Program on file streams**

Problem Statement:

Write a program that creates a file by reading and storing user input text. The program further reads the file and creates another file that contains all the text of the input file converted into uppercase.

CODE:

#include <iostream> #include <fstream> using namespace std; int main()

{

ofstream input("C:\Users\Prajwal\Desktop\\input.txt"); string s;

cout << "Enter the content to store in the file::"; getline(cin, s);

input << s; input.close(); ifstream read;

read.open("C:\Users\Prajwal\Desktop\\input.txt"); if (!read)

{

cout << "No such file"; exit(EXIT\_FAILURE);

}

ofstream output("C:\Users\Prajwal\Desktop\\output.txt"); char ch;

read >> noskipws; while (1)

{

read >> ch; if(read.eof())

break;

output << ((char)toupper(ch));

}

cout << "\nContent written in UpperCase Successfully!!"; read.close();

output.close(); return 0;

}

OUTPUT:

