ASSIGNMENT 3:

Name: Om Prasanna Kulkarni

PRN: 22510034

Batch: S3

Branch: CSE

Topic: Inheritance and Operator Overloading

1)Make a class named Fruit with a data member to calculate the number of fruits in a basket. Create two other class named Apples and Mangoes to calculate the number of apples and mangoes in the basket. Print the number of fruits of each type and the total number of fruits in the basket.

1. Source Code:

#include<iostream>

using namespace std;

class Fruit {

protected:

    int FruitNO;

public:

    void addFruits(int numFruits) {

      FruitNO = numFruits;

       cout << "Total number of fruits in the basket: " << FruitNO <<endl;

    }

};

class Apples : public Fruit {

public:

    int apples;

    void addApples(int numApples) {

        apples = numApples;

        cout << "The number of Apples are: " << apples << endl;

    }

};

class Mangoes : public Fruit {

public:

    int mangoes;

    void addMangoes(int numMangoes) {

        mangoes = numMangoes;

        cout << "The number of Mangoes are: " << mangoes << endl;

    }

};

int main() {

    int apples, mangoes;

    Apples obj1;

    Mangoes obj2;

    cout << "Enter the number of apples: ";

    cin >> apples;

    obj1.addApples(apples);

    cout << "Enter the number of mangoes: ";

    cin >> mangoes;

    obj2.addMangoes(mangoes);

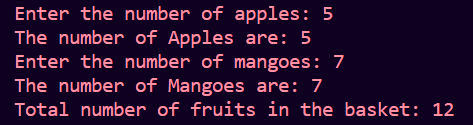
    int totalFruits = apples + mangoes;

    obj1.addFruits(totalFruits);

    return 0;

}

OOUTPUT :



2)Create a class named Shape with a function that prints "This is a shape". Create another class named Polygon inheriting the Shape class with the same function that prints "Polygon is a shape". Create two other classes named Rectangle and Triangle having the same function which prints "Rectangle is a polygon" and "Triangle is a polygon" respectively. Again, make another class named Square having the same function which prints "Square is a rectangle". Now, try calling the function by the object of each of these classes.

Source Code:

#include<iostream>

using namespace std;

class Shape{

public:

void printShape(){

    cout<<"This is a shape. "<<endl;

}

};

class Polygon:public Shape{

    public:

    void printShape(){

    cout<<"Polygon is a shape. "<<endl;

}

};

class Rectangle:public Polygon{

public:

void printShape(){

    cout<<"Rectangle is a polygon. "<<endl;

}

};

class Triangle:public Polygon{

public:

void printShape(){

    cout<<"Triangle is a polygon. "<<endl;

}

};

class Square:public Rectangle{

 public:

 void printShape(){

    cout<<"Square is a rectangle. "<<endl;

 }

};

int main(){

    Square s;

    s.printShape();

    Triangle t;

    t.printShape();

    Rectangle r;

    r.printShape();

    Polygon p;

    p.printShape();

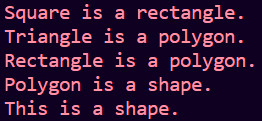
    Shape sh;

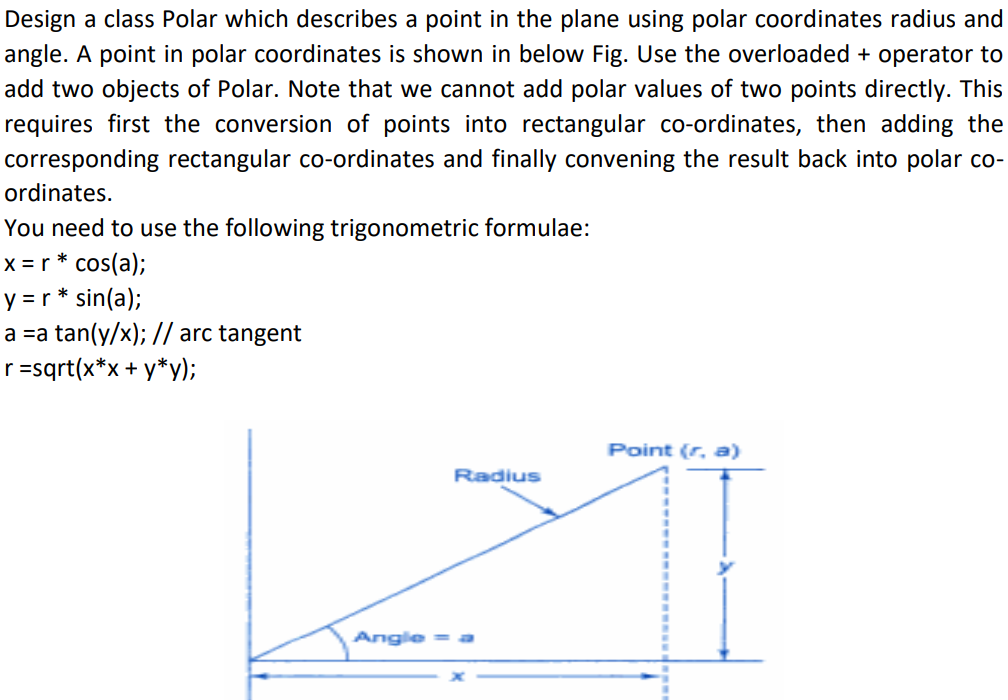
    sh.printShape();

    return 0;

}

OUTPUT:





Source Code:

#include<iostream>

#include<math.h>

using namespace std;

class Polar{

    double radius,angle;

    public:

    Polar(int r,float a){

         radius=r;

         angle=a;

    }

    double getRadius(){

        return radius;

    }

        double getAngle(){

        return angle;

    }

    void setRadius(double r){

        radius=r;

    }

    void setAngle(double a){

        angle=a;

    }

    void display(){

        cout<<"Result\n";

        cout<<"Polar: ("<<radius<<","<<angle<<")"<<endl;

    }

    Polar operator+(Polar &obj){

        double x1=radius\*cos(angle);

        double y1=radius\*sin(angle);

        double x2=obj.radius\*cos(obj.angle);

        double y2=obj.radius\*sin(obj.angle);

    double X=x1+x2;

    double Y=y1+y2;

    double radiusSum = sqrt(X \* X + Y \* Y);

    double angleSum = atan2(Y, X);

        return Polar(radiusSum, angleSum);

    }

};

int main(){

    double d1,d2;

    cout<<"Enter 1st Polar coordinates: "<<endl;

    cin>>d1>>d2;

    Polar p1(d1,d2);

    double b1,b2;

    cout<<"Enter 2nd Polar coordinates: "<<endl;

    cin>>d1>>d2;

    Polar p2(d1,d2);

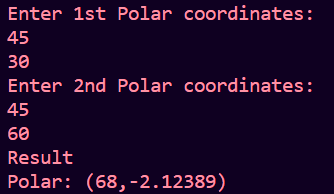
    Polar p3=p1+p2;

    p3.display();

    return 0;

}

OUTPUT:



4) . Define a class String. Use overloaded = = operator to compare two strings.

Source Code:

#include <iostream>

#include <cstring>

using namespace std;

class String {

private:

    char\* str;

public:

    String(const char\* s = "") {

        str = new char[strlen(s) + 1];

        strcpy(str, s);

    }

    String(const String&   t) {

        str = new char[strlen(  t.str) + 1];

        strcpy(str,   t.str);

    }

    ~String() {

        delete[] str;

    }

    String& operator=(const String&   t) {

        if (*this* == &  t) {

            return \**this*;

        }

        delete[] str;

        str = new char[strlen(  t.str) + 1];

        strcpy(str,   t.str);

        return \**this*;

    }

    bool operator==(const String&   t) const {

        return (strcmp(str,   t.str) == 0);

    }

    void display() const {

        cout << str;

    }

};

int main() {

    char input1[100], input2[100];

    cout << "Enter the first string: ";

    cin.getline(input1, 100);

    cout << "Enter the second string: ";

    cin.getline(input2, 100);

    String s1(input1);

    String s2(input2);

    String s3 = s1;

    String s4;

    s4 = s2;

    cout << "s1: ";

    s1.display();

    cout << endl;

    cout << "s2: ";

    s2.display();

    cout << endl;

    cout << "s3: ";

    s3.display();

    cout << endl;

    cout << "s4: ";

    s4.display();

    cout << endl;

    if (s1 == s2) {

        cout << "s1 and s2 are equal." << endl;

    } else {

        cout << "s1 and s2 are not equal." << endl;

    }

    if (s2 == s4) {

        cout << "s2 and s4 are equal." << endl;

    } else {

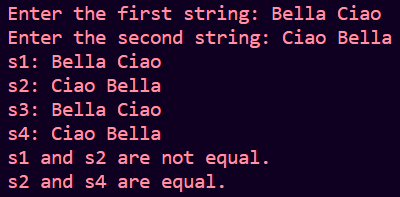
        cout << "s2 and s4 are not equal." << endl;

    }

    return 0;

}

OUTPUT:



5) Scenario: University System Your task is to design and implement a simple university system that has several roles such as professors, students, and administrative staff. All roles share some common attributes, but they also have their unique attributes and functions. Task Breakdown:

1. Common Base Class: Person Attributes:

• Name

• Age

• ID Methods:

• Display details

• Modify details

2. Derived Class: Student Inherits from: Person

• Additional Attributes:

• Major (e.g., Computer Science, Physics, Literature)

• GPA

• Methods:

• Register for a course

• View registered courses

• Calculate academic standing (e.g., based on GPA) 3. Derived Class: Professor Inherits from: Person Additional Attributes:

• Department (e.g., Physics, Computer Science)

• Courses taught Methods:

• Add a course to teach

• List courses being taught

• Advise a student (print student's details and give hypothetical advice based on GPA)

4. Derived Class: AdminStaff Inherits from: Person

• Additional Attributes:

• Job title (e.g., Registrar, Financial Aid Officer) Methods:

• Send announcement (print a hypothetical announcement to the console)

• Attend meeting (print a hypothetical meeting summary)

Make university class through which all functionality should be accessible. Note: Use private, protected and public access specifiers appropriately.

Source Code:

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Person

{

private:

    int age, ID;

    string name;

public:

    Person()

    {

        name = "";

        age = 0;

        ID = 0;

    }

    Person(string &ipname, int ipid, int ipage) : name(ipname), ID(ipid), age(ipage){};

    virtual void display()

    {

        cout << "Name is : " << name << endl

             << "ID : " << ID << " Age : " << age << endl;

    }

    void modify(string &name\_to\_modify, int age\_to\_modify)

    {

        name = name\_to\_modify;

        age = age\_to\_modify;

    }

};

class Student : public Person

{

private:

    string major;

    float cgpa;

    vector<string> regcourses;

public:

    Student(string ipname, int ipid, int ipage, string ipmajor) : Person(ipname, ipid, ipage), major(ipmajor){};

    void set\_cgpa()

    {

        cout << "Enter CGPA of Student : ";

        cin >> cgpa;

    }

    void display()

    {

        Person::display();

        cout << "Major is : " << major << endl

             << "CGPA : " << cgpa << endl;

    }

    void modify(string &name\_to\_modify, int age\_to\_modify)

    {

        Person::modify(name\_to\_modify, age\_to\_modify);

    }

    void view\_courses()

    {

        cout << "Student is registered for Courses : " << endl;

        for (auto it : regcourses)

        {

            cout << it << " ";

        }

        cout << endl;

    }

    void add\_courses()

    {

        int temp;

        string course;

        cout << "Enter Number of Courses to be added : ";

        cin >> temp;

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

        {

            cin >> course;

            regcourses.push\_back(course);

        }

    }

    void academic\_standing()

    {

        cout << "Student's Academic Standing : ";

        if (cgpa <= 4)

            cout << "Poor" << endl;

        else if (cgpa > 4 && cgpa < 7.5)

            cout << "Good" << endl;

        else if (cgpa >= 7.5 && cgpa <= 10)

            cout << "Excellent" << endl;

    }

};

class Professor : public Person

{

private:

    string department;

    vector<string> courses;

public:

    Professor(string ipname, int ipid, int ipage, string ipdep) : Person(ipname, ipid, ipage), department(ipdep){};

    void display()

    {

        Person::display();

        cout << "Department is : " << department << endl;

    }

    void addcourse(string course\_to\_add)

    {

        courses.push\_back(course\_to\_add);

    }

    void course\_info()

    {

        cout << "Courses taught by teacher are : " << endl;

        for (auto it : courses)

        {

            cout << it << " ";

        }

    }

};

class AdminStaff : public Person

{

private:

    string job;

public:

    AdminStaff(string ipname, int ipid, int ipage, string ipjob) : Person(ipname, ipid, ipage), job(ipjob){};

    void display()

    {

        Person::display();

        cout << "Job Title is : " << job << endl;

    }

    void announcement()

    {

        cout << "Announcement" << endl;

    }

    void meetsummary()

    {

        cout << "Summary of the meet." << endl;

    }

};

class University : public Person

{

private:

    vector<Person> Person\_info;

public:

    void addPerson(Person &personobj)

    {

        Person\_info.push\_back(personobj);

    }

    void display\_info()

    {

        Person::display();

    }

};

int main()

{

    string name, major, dep, job;

    int id, age;

    cout << "Enter Name of Student : ";

    cin >> name;

    cout << "Enter ID of Student : ";

    cin >> id;

    cout << "Enter Age of Student : ";

    cin >> age;

    cout << "Enter Major of Student : ";

    cin >> major;

    cout << endl;

    Student s(name, id, age, major);

    s.set\_cgpa();

    s.display();

    s.add\_courses();

    s.view\_courses();

    s.academic\_standing();

    cout << endl;

    cout << "Enter Name of Professor : ";

    cin >> name;

    cout << "Enter ID of Professor : ";

    cin >> id;

    cout << "Enter Age of Professor : ";

    cin >> age;

    cout << "Enter Department of Professor : ";

    cin >> dep;

    cout << endl;

    Professor p(name, id, age, dep);

    p.addcourse("Engineering Mathematics");

    p.display();

    p.course\_info();

    cout << endl;

    cout << "Enter Name of Staff Member : ";

    cin >> name;

    cout << "Enter ID of Staff Member : ";

    cin >> id;

    cout << "Enter Age of Staff Member : ";

    cin >> age;

    cout << "Enter Job Title of Staff Member : ";

    cin >> job;

    cout << endl;

    AdminStaff a(name, id, age, job);

    a.display();

    a.announcement();

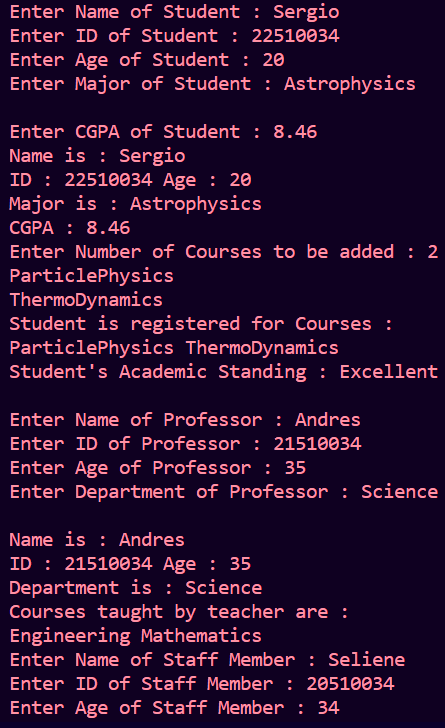
    a.meetsummary();

    cout << endl;

    return 0;

}

**OUTPUT:**

****