January 2025 CSE 108

Online 1: A1+A2

Time: 45 min Total Mark: 10

You need to simulate a **2D Geometric Modeling System** using basic Object-Oriented Programming (OOP) concepts in C++. The system should allow the creation of 2D points and simple polygons composed of such points.The polygon's perimeter and area should be computed using the standard Euclidean distance and Shoelace formula, respectively. **You can safely assume that the polygon will have no more than 10 vertices.**

Tasks

1. Create a class named `Point` with the following private member variables:
    - `double x` – x-coordinate of the point
    - `double y` – y-coordinate of the point
2. Implement the following public member functions in the `Point` class:
    - `void setCoordinates(double x, double y)` – sets the point's coordinates
    - `Point translate(double dx, double dy)` – returns a new point translated by `dx` and `dy`
    - `double cross(const Point& other)` – returns the cross product of this point and the other
    - `double getDistance(const Point& other)` – returns the Euclidean distance between this point and the other
    - `void display()` – prints the coordinates of the point
3. Implement the following constructors for the `Point` class:
    - Default constructor: `Point()`– initializes the point to (0, 0)
    - Parameterized constructor: `Point(double x, double y)`

4. Create a class named `Polygon` with the following private member variables:

    - `Point* vertices` – dynamically allocated array of points
    - `int numVertices` – number of vertices in the polygon

5. Implement the following public member functions in the `Polygon` class:

    - `void addVertex(Point p)` – adds a new vertex to the polygon. **You can safely assume that vertices added to the polygon will always be in the counterclockwise order.**
    - `double getPerimeter()` – calculates the perimeter using Euclidean distances
    - `double getArea()` – calculates the area using the Shoelace formula
    - `Polygon translate(double dx, double dy)` – returns a new polygon with all vertices translated
    - `void display()` – displays all vertices, the perimeter, and the area

6. Implement the following constructors and special functions for the `Polygon` class:

    - Default constructor: initializes an empty polygon
    - Parameterized constructor: `Polygon(Point* vertices, int n)` – creates a polygon with `n` vertices
    - Copy constructor
    - Destructor

7. Please carefully see the sample main function to understand the output format. You can copy the main function directly to your code.

8. You are NOT allowed to define any public member functions other than the ones mentioned above for either class, not even getters and setters other than the ones mentioned.

Shoelace Formula Reminder (For Area of a Simple Polygon):

$$\text{For a simple polygon with } n \text{ vertices: } (x_1, y_1), (x_2, y_2), ..., (x_n, y_n),$$
$$\text{Area} = \frac{1}{2}|(x_1y_2 + x_2y_3 + \cdots + x_ny_1) - (y_1x_2 + y_2x_3 + \cdots + y_nx_1)|$$

Sample `main()` Function:

```
int main() {
    Point p1(0, 0);
    Point p2(4, 0);
    Point p3(4, 3);
    Point p4(0, 3);

    Point vertices[] = {p1, p2, p3};
    Polygon triangle(vertices, 3);

    Polygon translatedTriangle = triangle.translate(2, 1);

    cout << "Original Triangle:\n";
    triangle.display();

    cout << "\nTranslated Triangle:\n";
    translatedTriangle.display();

    Polygon square = triangle;
    square.addVertex(p4);
    cout << "\nSquare:\n";
    square.display();

    return 0;
}
```

Expected Output:

```
Original Triangle:
No. of Vertices: 3
(0, 0)
(4, 0)
(4, 3)
Perimeter: 12, Area: 6

Translated Triangle:
No. of Vertices: 3
(2, 1)
(6, 1)
(6, 4)
Perimeter: 12, Area: 6

Square:
No. of Vertices: 4
(0, 0)
(4, 0)
(4, 3)
(0, 3)
Perimeter: 14, Area: 12
```