**Advance Programming**

**CSCI 251**

**Assignment 2**

**Name: Jeslyn Ho Ka Yan**

**SIM ID/ UOW ID: 10241485/ 8535383**

**Tutorial Group Number: Group 6**

**Table of Content**

1 Requirements of the Program 3

2 Program Design 3

3 Implementation Summary 4

3.1 1. ShapeTwoD Base Class 4

3.2 2. Circle Class 4

3.3 3. Rectangle Class 4

3.4 Square Class 4

3.5 Cross Class 5

3.6 Main Program 5

4 Reflections 6

# Requirements of the Program

The objective of this program is to allow users to:

1. Input sensor data for various 2D shapes (Circle, Rectangle, Square, and Cross).
2. Compute and display the area of all entered shapes.
3. Generate and print a detailed report of all shapes, including their perimeter and inner points.
4. Sort shapes based on area, special type, or a combination of both.

This program is developed as a menu-driven console application and should provide user-friendly interactions.

# Program Design

* **High-Level Design**
* **Main Class**: Contains the menu interface and handles user inputs.
* **ShapeTwoD Base Class**: Serves as an abstract base class for all specific shapes. Defines common functionalities such as computing area and determining inner or perimeter points.
* **Derived Classes**:
  + **Circle**: Represents a circle with a center and radius.
  + **Rectangle**: Represents a rectangle defined by two diagonally opposite corners.
  + **Square**: Inherits from Rectangle with equal width and height.
  + **Cross**: Represents a cross shape defined by 12 vertices.

# Implementation Summary

## ShapeTwoD Base Class

* Defined in ShapeTwoD.h and ShapeTwoD.cpp.
* Provides abstract methods:
  + **computeArea:** Calculates the shape's area.
  + **generatePerimeterPoints:** Lists all points on the shape's perimeter.
  + **generateInnerPoints:** Lists all points inside the shape.
  + **toString:** Generates a detailed string representation of the shape.

## Circle Class

* Defined in Circle.h and Circle.cpp.
* Specific Attributes: Center coordinates and radius.
* Key Functions:
  + Area computation using.
  + Generating points on the perimeter and inside the circle.

## Rectangle Class

* Defined in Rectangle.h and Rectangle.cpp.
* Specific Attributes: Top-left and bottom-right coordinates.
* Key Functions:
  + Area computation as width height.
  + Generating perimeter and inner points.

## Square Class

* Defined in Square.h and Square.cpp.
* Inherits from Rectangle.
* Overrides methods to ensure side lengths are equal.

## Cross Class

* Defined in Cross.h and Cross.cpp.
* Specific Attributes: Array of 12 vertices defining the cross.
* Key Functions:
  + Area computation using the Shoelace formula.
  + Generating perimeter and inner points.

## Main Program

* Defined in A2\_Jeslyn.cpp.
* Implements a menu with the following options:

1. Input sensor data for shapes.
2. Compute area for all shapes.
3. Generate a report of shapes.
4. Sort shapes by area or special type.
5. Exit the program.

* Uses std::vector to store shape objects dynamically.

# Reflections

**Assumptions**

* The program assumes valid user inputs for shape coordinates and dimensions.
* Shape names and types (e.g., "Circle", "NS/WS") are case-insensitive.
* Sorting behavior adheres strictly to the specified criteria (area, special type).

**Challenges**

* Ensuring accuracy of geometric calculations, especially for complex shapes like Cross.
* Designing a user-friendly menu interface.
* Handling edge cases such as overlapping vertices or incorrect inputs.

**What Could Be Improved**

* Extend support for additional shapes (e.g., Triangle, Ellipse).
* Add graphical visualization for shapes instead of textual reports.
* Incorporate persistent storage for shape data.

**Enhancements in Future**

* GUI-based implementation for better user interaction.
* Advanced algorithms for faster shape analysis and computations.
* Real-time data validation to prevent user input errors.

**Learnings**

* Application of OOP principles such as inheritance and polymorphism.
* Effective use of STL containers like std::vector.
* Importance of modular design and reusability in software development.