FINAL EXAMINATION

Course: **CSC10003 OBJECT-ORIENTED PROGRRAMMING**

Time: **100** minutes (**10 points**) Term: 1 – Academic year: **2024-2025**

(Notes: one A4 sheet of document (handwritten or printed text on both sides) is allowed)

Question 1 (1 point): Tell the differences between "`struct`" and "`class`" in C++

Question 2 (2 points): Assume all necessary libraries are included, read the C++ code below and answer the following questions:

```
01  class Figure {
02   protected:
03    string _name;
04   public:
05    virtual void showFigure() {
06     cout << _name << endl;
07    }
08    virtual double calcArea() = 0;
09  };
10  class Point {
11   private:
12    int _x, _y;
13   public:
14    Point(int x, int y) {
15     cout << "1st point constructor" << endl;
16     _x = x;
17     _y = y;
18    }
19    Point(const Point& I) {
20     cout << "2nd point constructor" << endl;
21     _x = I._x;
22     _y = I._y;
23    }
24    void setX(int x) { _x = x; }
25    void setY(int y) { _y = y; }
26    int getX() { return _x; }
27    int getY() { return _y; }
28    ~Point() {
29      cout << "Point destructor" << endl;
30    }
31  };
```

```
32  class Circle: public Figure {
33   private:
34    Point* _I; double _R;
35   public:
36    void setI(const Point& I) {
37     if (_I == nullptr) _I = new Point(I);
38     else *(_I) = I;
39    }
40    void setR(double R) {
41     if(R >= 0) _R = R;
42    }
43    double getR() { return _R; }
44    double calcArea() { return 3.14 * _R * _R; }
45    Circle() {
46     cout << "Circle constructor" << endl;
47     _name = "Circle"; _I = nullptr; _R = 0;
48    }
49    ~Circle() {
50     if (_I != nullptr)delete _I;
51     cout << "Circle destructor" << endl;
52    }
53    void showFigure() {
54      cout << _name << ": " << calcArea() << endl;
55    }
56  };
57  void main() {
58   Circle* c = new Circle();
59   Point I(1, 2);
60   c->setI(I); c->setR(2);
61   c->showFigure();
62   delete c;
63  }
```

a) What are printed on the screen when compiling and executing the above program? (1 point)

b) Explain the order of execution of the program (1 point)

Question 3 (3 points): Write a C++ program to implement a Complex class to represent complex numbers. The class should:

1. Have **two private data members**: real (for the real part) and imag (for the imaginary part).
2. Provide **constructors** for:
    - Default initialization (0 + 0i),
    - Parameterized initialization.
3. Overload the following operators:

- + to add two complex numbers,
- - to subtract two complex numbers,
- * to multiply two complex numbers.

4. Overload the **stream operators** << and >>:
   - >> should allow the user to input the real and imaginary parts of a complex number.
   - << should display a complex number in the format: a + bi.

Question 4 (4 points)**:** Given this main function

```
#include <iostream>
#include <vector>

void main() {
    std::vector<IShape*> shapes = {
        new Rectangle(10, 6),
        new Square(5),
        new Rectangle(8, 5),
        new Square(3)
    };

    for (const IShape* shape : shapes)    {
        std::cout << shape->toString() << "\n";
    }
}
```

Sample output

```
Rectangle Width=10, Height=6
Square Side=5
Rectangle Width=8, Height=5
Square Side=3
```

Requirements.

1. Draw class diagram.
2. Declare and implement all the classes.
3. What if in the constructor of the two classes Rectangle and Square, the parameters are negative? This would lead to the failure of initialization.
   Propose a solution to this problem. Redraw the class diagram or rewrite the affected block of code if needed.
4. What if we want to display these shapes into **solid** and **hollow** shapes, with customizable display character like *, +, @, -?

```
**********
**********
**********
**********
**********
**********

+++++
+   +
+   +
+   +
+++++

@@@@@@@@
@@@@@@@@
@@@@@@@@
@@@@@@@@
@@@@@@@@

---
-  -
---
```

This line of code will not be relevant.

```
std::cout << shape->toString() << "\n";
```

Propose a solution to this problem. Redraw the class diagram or rewrite the affected block of code if needed.

---End---