Exercise 7:

Submit your solutions (source code) to the questions below by email to your instructor and TA(s) by Monday, November 19th (16:30).

Question 1: Two dimensional shape interface (65 points).

This question will test your understanding of dynamic polymorphism, virtual functions and abstract base classes. You have to write three classes: Rectangle and Circle that implement a common interface Shape2D.

Consider the interface (or abstract base class) Shape2D. This interface exports the following methods:

- string get_name() const: returns a string corresponding to the name of the primitive (e.g. cirlce, ellipse, ...)
- double compute_perimeter() const: returns the perimeter of the shape
- double compute_area() const: returns the area of the shape

Shape2D is an interface (or abstract base class), therefore all its methods are pure virtual. Define the interface Shape2D in a file named Shape2D.h.

```
// Shape2D.h

#include <string>

#ifndef SHAPE2D_H
#define SHAPE2D_H

class Shape2D {
 // COMPLETE
};

#endif // SHAPE2D_H
```

The next step is to implement the classes: Rectangle and Circle. These classes implement the interface Shape2D.

The constructor for Rectangle is:

- Rectangle(left_corner, width, height) that specifies the leftmost corner, the width and height of the rectangle

The constructor for Circle is:

- Circle(center, radius) that specifies the center and radius of the circle

All the quantities above have type double (or array of double for vectors).

Please create the files: "Rectangle.h", "Rectangle.cpp", "Circle.h" and "Circle.cpp". Header files will contain the classes definition. The code for the methods as well as the constructors

(and destructor) should go in the .cpp files.

To test your code, please type (and use) the following code in a file "test_Shape2D.cpp"

```
#include <iostream>
#include <string>
#include "Shape2D.h"
#include "Circle.h"
#include "Rectangle.h"

void compute_and_print_shape_info(const Shape2D& s) {
  std::cout << "Shape: " << s.get_name() << std::endl;
  std::cout << "Perimeter: " << s.compute_perimeter() << std::endl;
  std::cout << "Area: " << s.compute_area() << std::endl;
}

int main(void) {
 double center[2] = {0.0, 0.0};
 Circle c(center, 1.0);
 double left_corner[2] = {-1.0, -1.0};
 Rectangle r(left_corner, 4.0, 2.0);

 compute_and_print_shape_info(c);
 compute_and_print_shape_info(r);

 return 0;
}
```

Question 2: Polygons (35 points).

In addition to rectangles and circles, we want to handle polygons. A polygon is defined as the area enclosed by a closed curve made of linear segments.

A polygon is constructed from an integer "n" (corresponding to the number of points or vertices) and a list of 2d points "Pi = (xi,yi)" (xi and yi are the coordinates of the point along the x and y axis). We use the convention that Pn = P0 (the last vertex of the chain is the starting vertex of the chain) in order to have a closed loop of segments. The perimeter is straightforward to compute as the sum of the edges length. The area can be computed by the following formula: 2 * Area = Sum_{i=0}^{n-1} x_i y_{i+1} - y_i x_i{i+1}.

Please create the files: "Polygon.h" and "Polygon.cpp" and implement the class Polygon. To test your code please modify the file "test_Shape2D.cpp" from the question 1 as follow (only the lines delimited by the comment "new code" need to be added):

```
#include <iostream>
#include <string>
#include "Shape2D.h"
#include "Circle.h"
#include "Rectangle.h"

// new code for testing Polygon:
#include "Polygon.h"

void compute_and_print_shape_info(const Shape2D& s) {
  std::cout << "Shape: " << s.get_name() << std::endl;
  std::cout << "Perimeter: " << s.compute_perimeter() << std::endl;
  std::cout << "Area: " << s.compute_area() << std::endl;
}
```

```
int main(void) {
 double center[2] = {0.0, 0.0};
 Circle c(center, 1.0);
 double lett_corner[2] = {-1.0, -1.0};
 Rectangle r(left_corner, 4.0, 2.0);

 // new code for testing the class Polygon:
 int n = 4;
 double x[4] = {2.0, 2.0, -2.0, -2.0};
 double y[4] = {-1.0, 1.0, 1.0, -1.0};
 Polygon p(n, x, y);
 // end new code

 compute_and_print_shape_info(c);
 compute_and_print_shape_info(r);

 // new code for testing Polygon:
 compute_and_print_shape_info(p);
 // end new code

 return 0;
}
```