Program 1 **[35 Marks]**

You are given an incomplete program source code named **`program1.cpp`**. The program is intended to perform array manipulations consisting of adding and removing elements to/from an array. A class named `Array` has been defined in the program for this purpose. The array is specified such that it can only hold up to three elements. The elements are stored in the attribute `data`, whereas the number of elements currently held by the array is kept in the attribute `count`. When an element is added, `count` is increased by 1, and it is decreased by 1 if an element is removed. The class needs to provide a bound-error checking whereby elements cannot be added anymore when the array is already full. Similarly, removing an element cannot be accomplished if the array is empty. Also, accessing an element is allowable only if a valid index is used.

Complete the program according to the tasks stated in the program source code. The tasks are labeled as Task 1 to Task 9.

Figure 1.1 to 1.2 show example runs of the complete program for three types of cases. Note that, the **bold** texts indicate user inputs entered from the keyboard.

```
Number 11 has been added. Current number of element = 1
Number 22 has been added. Current number of element = 2

Enter the index of the element you want to display => 0
Index: 0, Element: 11

Number 33 has been added. Current number of element = 3
The array is full.

An element has been removed. Current number of element = 2
An element has been removed. Current number of element = 1
An element has been removed. Current number of element = 0
The array is empty
```

Figure 1.1: The user has entered a valid index for the element to be displayed. In this case, the value of the corresponding element will be displayed.

```
Number 11 has been added. Current number of element = 1
Number 22 has been added. Current number of element = 2

Enter the index of the element you want to display => -2
Error! You have entered a negative index.

Number 33 has been added. Current number of element = 3
The array is full.

An element has been removed. Current number of element = 2
An element has been removed. Current number of element = 1
An element has been removed. Current number of element = 0
The array is empty
```

Figure 1.2: The user has entered a negative index. In this case, an error occurs as the array should start at the index 0.

```
Number 11 has been added. Current number of element = 1
Number 22 has been added. Current number of element = 2

Enter the index of the element you want to display => 2
Error! You have entered index value of 2
 while the current number of elements is  2

Number 33 has been added. Current number of element = 3
The array is full.

An element has been removed. Current number of element = 2
An element has been removed. Current number of element = 1
An element has been removed. Current number of element = 0
The array is empty
```

Figure 1.3: The user has entered an index that is larger than the valid value the array currently has. In this case, the array is currently holding only two elements. That means, the valid index should be 0 or 1. Thus, an error occurs when the user is trying to access the element at index 2 (i.e., the third element)

Program 2 **[65 Marks]**

Rectangles and circles are most common geometrical shapes. A rectangle can be defined by two points, i.e., the top left and bottom right vertices. Whereas, a circle is defined by its center point and radius. Further, the area of a rectangle is obtained from its width and height, and the area of a circle is based only on its radius. Figure 2.1 and 2.2 show how to determine the area of a rectangle and a circle, respectively.

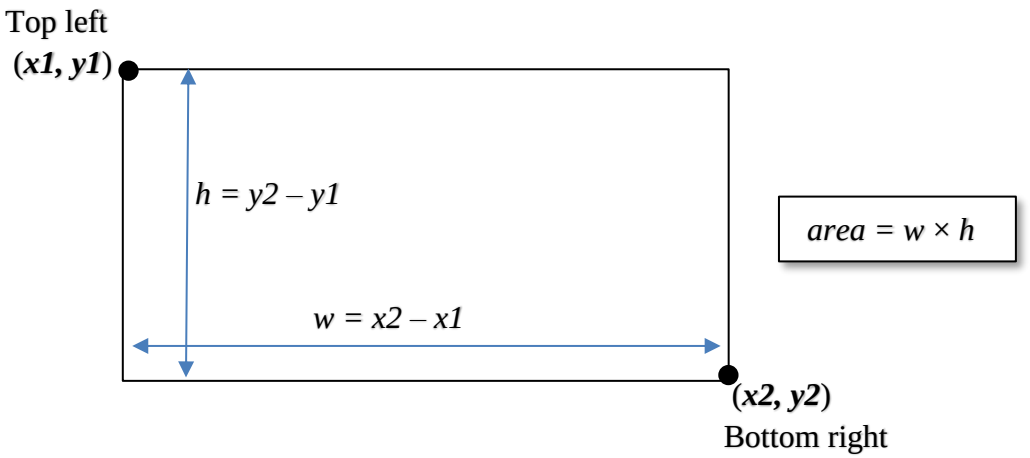

Figure 2.1: Formula to obtain the width (*w*), height (*h*) and area of a rectangle.

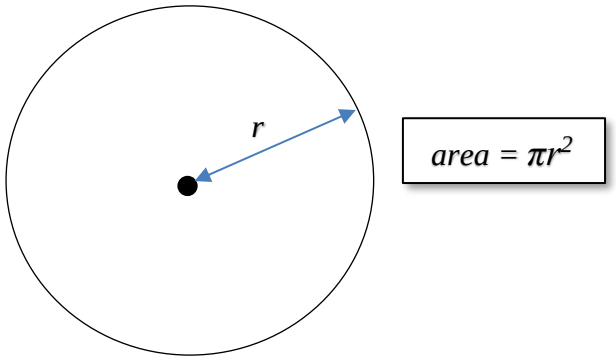

Figure 2.2: Formula to obtain the area of a circle with radius *r*. Note that, use $\pi=3.1415$.

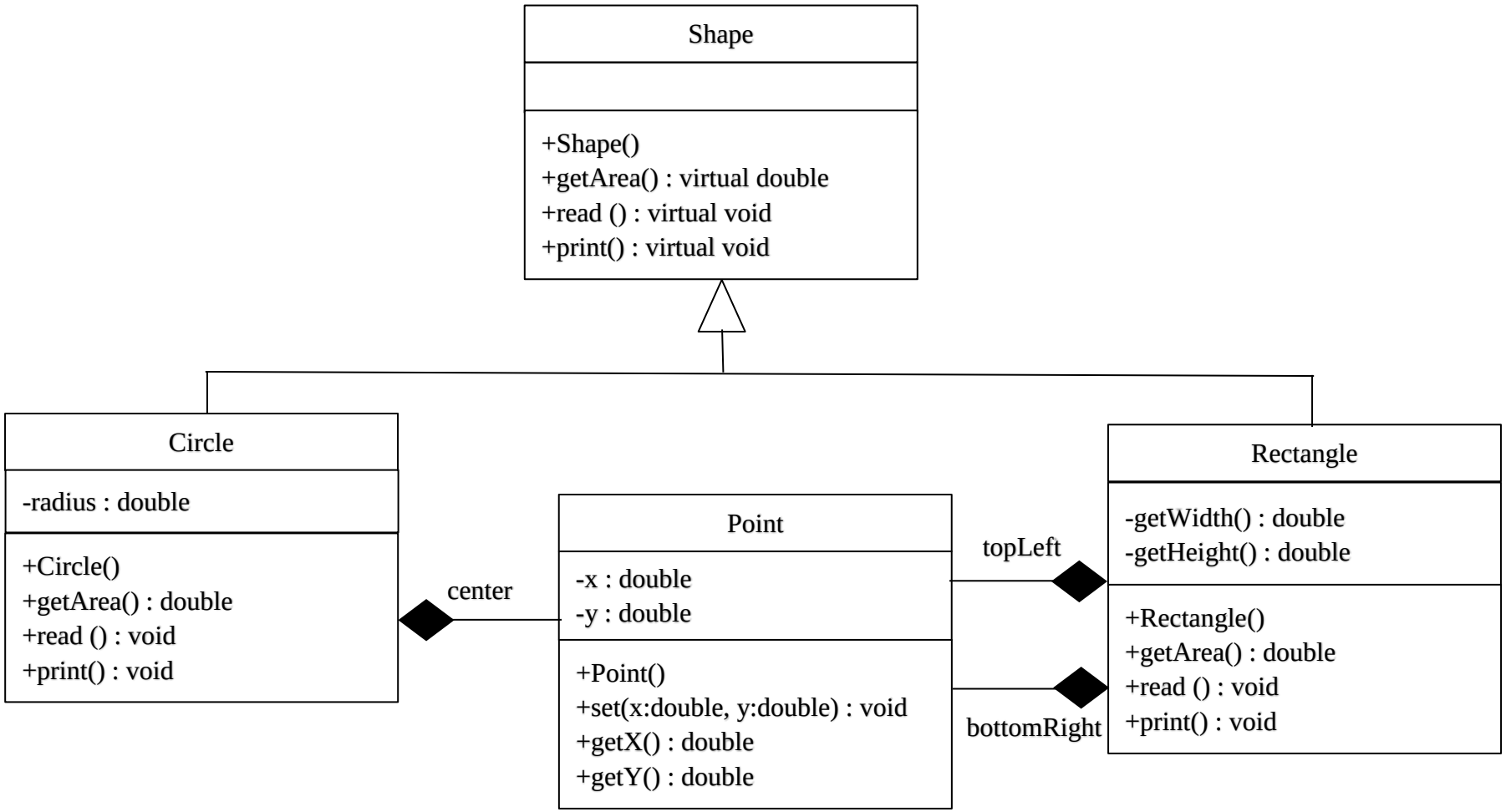

Figure 2.3: The class diagram

Given a class diagram consisting of three main classes, `Shape`, `Circle` and `Rectangle`, and their relationship, in Figure 2.3. Below are some important notes regarding the classes:

- The methods `getArea()` in class `Rectangle` and `Circle` are meant to calculate the area of the circle and rectangle, respectively. The formula to calculate the area are given in Figure 2.1 and 2.2.
- The methods `read()` in the classes will be used to read the data of the shape from the keyboard. As for the class `Rectangle` the data to read will be the coordinates of the top left and bottom right corners. Whereas, for the circle, the data to read are the coordinates of the center and the radius.
- The methods `print()` in the classes are meant to print shape's data onto the screen. As for the class `Rectangle` this method will be printing the top left and bottom right corners, the width and height, as well as the area of the rectangle. Whereas, for the circle, the data to print are the coordinates of the center and the radius, as well as the area of the circle.

Write a C++ program to implement all the classes given in Figure 2.3. Your implementation needs to apply several object-oriented programming concepts including **compositions, inheritances and polymorphisms**.

Then, use the classes to create a program that calculates areas of a list of shapes consisting of rectangles and circles. Your program needs to use only a single array to hold the shapes.

Also, your program needs to provide the user a menu-driven interaction with the following options.

Menu Options	Description
1. Add a shape	To insert a new shape (a circle or rectangle) into the array.
2. Print all shapes	To print the information of all the shapes.
3. Calculate total area	To calculate and print the total area of all the shapes.
4. Exit	To end the program.

Figure 2.4 shows the expected result of your program. Note that, all the interactions shown in the figure are continuous in a single run. Note also that, the **bold** texts indicate input entered by the user. The assessment criteria are given in Table 2.1.

Interaction 1: The user chooses option 1 to add a circle.

```
==========[MENU]============

1. Add a shape
2. Print all shapes
3. Calculate total area
4. Exit

Enter your choice => 1

What type of shape you want to enter?
     1. Circle
     2. Rectangle

Your choice => 1

Enter the center coordinates of the circle (x y)=> 0 0
Enter the circle's radius=> 10
```

Interaction 2: The user chooses option 1 to add a rectangle.

```
==========[MENU]============

1. Add a shape
2. Print all shapes
3. Calculate total area
4. Exit

Enter your choice => 1

What type of shape you want to enter?
     1. Circle
     2. Rectangle

Your choice => 2
1
Enter the top left corner coordinates of the rectangle (x y) => 20 30
Enter the bottom right corner coordinates of the rectangle (x y) => 30 20
```

Interaction 3: The user chooses option 1 to add another circle.

```
==========[MENU]============

1. Add a shape
2. Print all shapes
3. Calculate total area
4. Exit

Enter your choice => 1

What type of shape you want to enter?
     1. Circle
     2. Rectangle

Your choice => 1

Enter the center coordinates of the circle (x y)=> 50 50
Enter the circle's radius=> 50
```

Interaction 4: The user chooses option 2 to print the information of all the shapes that he/she has entered.

```
==========[MENU]============

1. Add a shape
2. Print all shapes
3. Calculate total area
4. Exit

Enter your choice => 2

Shape #1
Circle's center: X=0  Y=0
Circle's radius =10
Circle's area   =314.15

Shape #2
Rectangle's top left corner: X=20  Y=30
Rectangle's bottom right corner: X=30  Y=20
Rectangle's width  = 10
Rectangle's height = 10
Rectangle's area   = 100

Shape #3
Circle's center: X=50  Y=50
Circle's radius =50
Circle's area   =7853.75
```

Interaction 5: The user chooses option 3 to obtain the total area of all the shapes.

```
==========[MENU]============

1. Add a shape
2. Print all shapes
3. Calculate total area
4. Exit

Enter your choice => 3

Total Area= 8267.9
```

Interaction 6: The user chooses option 4 to end the program.

```
==========[MENU]============

1. Add a shape
2. Print all shapes
3. Calculate total area
4. Exit

Enter your choice => 4
```

Figure 2.4: An example run of the program.

Table 2.1: Assessment Criteria

Item	Criteria	Marks
A	The program can run with appropriate input and output.	3
	Using an appropriate structure for the program, including all required header files are included, the function main is properly written, and using proper indentations.	2
B	**Class definitions:**	
	`Point`	6
	`Shape`	3.5
	`Rectangle`	10
	`Circle`	7
C	**Implementation of OOP Concepts:**	
	Compositions	2
	Inheritances	2
	Polymorphisms	15.5
D	**The main program:**	
	Using a menu-driven interaction	2
	Using a single array to hold the list of shapes	3
	Adding a circle into the array	1
	Adding a rectangle into the array	1
	Printing all the shapes	2
	Calculating the total area	4
	Ending the program	1
	Total	**65**