CONFIDENTIAL

SCHOOL OF COMPUTING
Faculty of Engineering

UNIVERSITI TEKNOLOGI MALAYSIA

FINAL EXAMINATION SEMESTER I, 2018/2019

PAPER II (PRACTICAL)

SUBJECT CODE	**:**	**SCSJ1023**
SUBJECT NAME	**:**	**PROGRAMMING TECHNIQUE II**
SECTION	**:**	**1 and 2**
TIME	**:**	**(3 HOURS)**
DATE/DAY	**:**	
VENUE	**:**	

INSTRUCTIONS :

- This is a **CLOSED-BOOK** test. References to any resources by any means are strictly prohibited.
- The test consists of TWO (2) questions. **Answer all questions.**

MATERIAL FOR THE TEST:

- You are provided template programs for both questions, **program1.cpp** and **program2.cpp**.
- Download the material file (**scsj1023_xm-paper2.zip**) from **e-learning**. The file is password-protected. The password will be given later.
- **IMPORTANT NOTES: Do not edit the code directly** from the compression software. You **MUST extract** it first into your local hard drive.

SUBMISSION PROCEDURE:

- You must stop doing the test when the time is up and wait at the back.
- The submission will be done **offline** in batches (4 students at a time) under the instructor's supervisions.
- Only the source code files are required for the submission, i.e. **program1.cpp** and **program2.cpp**.

This question paper consists of EIGHT (8) printed pages excluding this page.

Program 1 **[35 Marks]**

Use the provided template program, **program1.cpp** to answer this question.

Consider the class diagram in Figure 1 which illustrates the data model for part-time students. A part-time student is a student who is also an employee for a company. The student can only register for subjects up to the maximum credit hours allowed. Besides, an advisor can be appointed for the student. Note that an advisor might be assigned to more than one students.

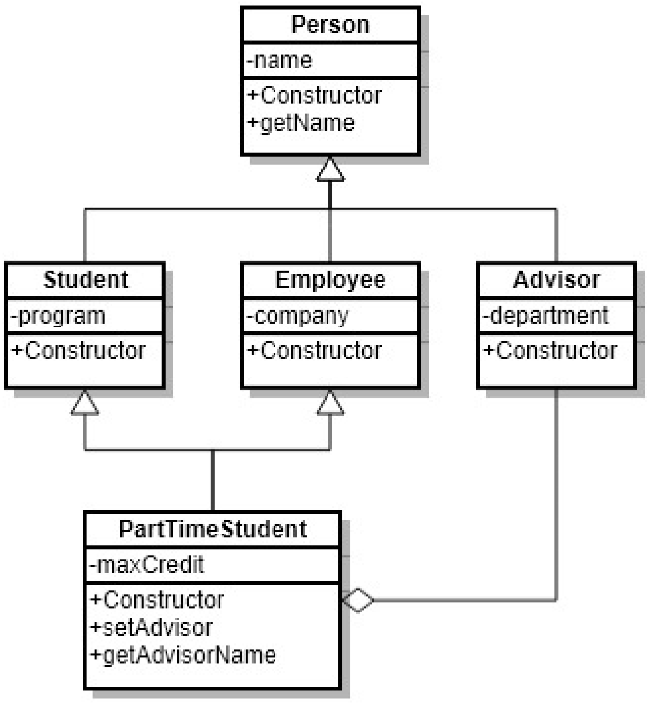

Figure 1: Class diagram for part-time students

Based on the class diagram in Figure 1, complete the given program by implementing all the classes. Regarding this, add the attribute (or attributes) and define a constructor for each class (other methods are not required). Use the constructor header as provided in the given program.

As for the class `PartTimeStudent`, add two more methods:

- `setAdvisor`, to appoint an advisor to a student.
- `getAdvisorName`, to get the name of the advisor. This method should also consider the case that the student has not yet had an advisor.

Note that, the implementation for the class `Person` has been given. Do nothing on this class. Also, do nothing on the main function.

The program will be assessed based on the criteria shown in Table 1. You must also ensure that your program is able to compile.

Table 1: Assessment criteria for Program 1

Item	Criteria	Marks
Task 1	Implementation of class `Student`	4
Task 2	Implementation of class `Employee`	4
Task 3	Implementation of class `Advisor`	4
Task 4	Implementation of class `PartTimeStudent`	23
	Total	**35**

Program 2 **[65 Marks]**

Use the provided template program, **program2.cpp** to answer this question.

Circles and rectangles are among the most common geometric shapes. Each shape has a location point consisting of coordinates *x* and *y*. For each circle, the point is defined by its center, while for each rectangle by its lower left corner. Furthermore, the size of each shape is defined by the radius (for the circle) and width and length (for the rectangle) as illustrated in Figure 2.

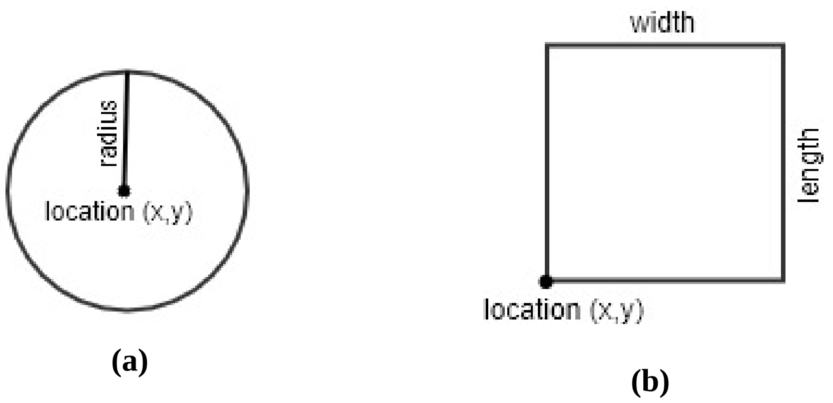

Figure 2: Representations of a circle (a) and rectangle (b).

The area of each shape is then calculated as:

$$\text{Circle's area} = \pi \text{ x } radius^2, \quad \text{where } \pi = 3.14$$

$$\text{Rectangle's area} = width \text{ x } length$$

Using the concept of polymorphism, inheritance and composition, write a program to manage a list of geometric shapes consisting of circles and rectangles. In the program, define a class named `List` to hold the list of shapes. The class has the following methods:

- `insert`, to add a shape to the list.
- `edit`, to modify the information of an existing shape in the list. The shape to be edited is determined by its index.
- `print`, to print the information of all the shapes consisting of the type, location, size and area of each shape, and the average location of the shapes. The average location is calculated by dividing the sum of all the shapes' location points with the number of shapes.

The program also needs to define other classes named `Point`, `Shape`, `Circle`, and `Rectangle`. In this regard, provide only the default constructor for each class and add other attributes and methods whenever necessary. Note that, the implemantaion for the class `Point` has already been done with two overloaded operators and some other methods. Do nothing on this class.

Furthermore, the program must be able to handle two illegal actions below:

- the user tries to add a new shape, but no more space left in the list.
- the user tries to edit a shape but he or she has entered an invalid array index. For example, the number of shapes in the list is three but the user tries to edit the fourth one, or the user might enter a negative value for the index.

Implement this error handling using exceptions. In this case, define two exception classes named `OutOfSpace` and `InvalidIndex`, respectively.

Figure 3 shows the expected result of the program. All the screens shown here are continuous in a single run. The **bold** texts indicate user inputs. Meanwhile, Table 2 shows the assessment criteria for this question.

Screen 1: The user chooses option 1 and option 2 (several times) to enter circles and rectangles.

```
========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 1


Enter a circle

Center: Enter x and y => 1 1
Radius => 1

========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 2

Enter a rectangle

Lower Left Corner: Enter x and y => 2 2
width and length => 2 2

========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit


Choose an operation [1-5] => 1

Enter a circle

Center: Enter x and y => 1.2 3.4
Radius => 5.5

========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 1

Enter a circle

Center: Enter x and y => 5 6
Radius => 7
```

```
========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 2


Enter a rectangle

Lower Left Corner: Enter x and y => 9.5 8.5
width and length => 3.2 6.8
```

Screen 2: The user chooses option 3 to print the list of shapes.

```
========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 3

Index   Shape Type      Location        Size        Area
-----   ----------      ---------       ----        ----
0       Circle          (1,1)           1           3.14
1       Rectangle       (2,2)           2x2         4
2       Circle          (1.2,3.4)       5.5         94.985
3       Circle          (5,6)           7           153.86
4       Rectangle       (9.5,8.5)       3.2x6.8     21.76

Average location:       (3.74,4.18)
```

Screen 3: The user chooses option 4 (several times) to edit existing shapes.

```
========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 4

Enter the index => 0
Enter a circle

Center: Enter x and y => 10 10
Radius => 10
```

```
========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 4

Enter the index => 1
Enter a rectangle

Lower Left Corner: Enter x and y => 20 20
width and length => 20 20
```

Screen 4: The user chooses option 3 to print the updated list.

```
========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 3

Index   Shape Type      Location        Size       Area
-----   ----------      --------        ----       ----
0       Circle          (10,10)         10         314
1       Rectangle       (20,20)         20x20      400
2       Circle          (1.2,3.4)       5.5        94.985
3       Circle          (5,6)           7          153.86
4       Rectangle       (9.5,8.5)       3.2x6.8    21.76

Average location:       (9.14,9.58)
```

Screen 5: The user chooses option 4 (several times) to edit shapes, however with invalid array indices.

```
========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 4

Enter the index => -1
** Error: Invalid Index
```

```
========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 4

Enter the index => 5
** Error: Invalid Index
```

Screen 6: The user chooses option 1 or 2 to add another circle or rectangle, however the list is already full (assuming the capacity of the list is 5)

```
========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit shape
5. Exit

Choose an operation [1-5] => 1

Enter a circle

Center: Enter x and y => 1

Radius => 1
** Error: Out of space

========== Menu ==========
1. Add circle
2. Add rectangle
3. Print list
4. Edit item
5. Exit

Choose an operation [1-5] => 2

Enter a rectangle

Lower Left Corner: Enter x and y => 2 2
width and length => 2 2
** Error: Out of space
```

Figure 3: Examples run of the program.

Table 2: Assessment criteria for Program 2

Item	Criteria	Marks
A	The code is properly written including using proper indentations and naming conventions.	1
B	**Class implementation:**	
	`Shape`	7
	`Circle`	10
	`Rectangle`	10
	`List`	14
C	**Exception implementation:**	
	Defining exception classes	2
	Handling errors using exceptions	5
D	**Main function:**	
	Adding a circle	3
	Adding a rectangle	3
	Printing the list	1
	Editing the s hape	2
E	**Program run:** The program must be able to run to be assessed for this part. Otherwise, a zero mark will be given automatically.	
	User input: to enter a circle, is working	1
	User input: to enter a rectangle, is working	1
	The screen output	2
	User input: to edit a shape, is working	1
	The exception to handle "out of space" is working	1
	The exception to handle "invalid index" is working	1
	Total	**65**