CONFIDENTIAL

UNIVERSITI TEKNOLOGI MALAYSIA

FINAL EXAMINATION
PAPER II (PRACTICAL)
SEMESTER II 2017/2018

SUBJECT CODE	**: SCSJ1023**
SUBJECT NAME	**: PROGRAMMING TECHNIQUE II**
YEAR/COURSE	**: 1 (SCSB/SCSJ /SCSP/SCSR / SCSV)** **2 (SCSR/SCSV)**
TIME	**: 2:30 PM – 5:30 PM (3 HOURS)**
DATE	**: 6 JUNE 2018**
VENUE	**: MPK 1-10, CGMTL, CASE, N28, FC**

INSTRUCTIONS TO THE STUDENTS:

This test book consists of 2 programming questions:

Program 1	35 Marks
Program 2	65 Marks
TOTAL	**100 Marks**

ANSWER ALL QUESTIONS.

MATERIAL FOR THE TEST:

- You are provided with two template source code files, **`program1.cpp`** and **`program2.cpp`**.
- Download the files (compressed in a RAR file named **`scsj1023-paper2.rar`**) from e-learning.
- **IMPORTANT:** You **MUST extract** the RAR file into the local hard drive of your computer. **Do not edit the code directly** from the WinRAR.

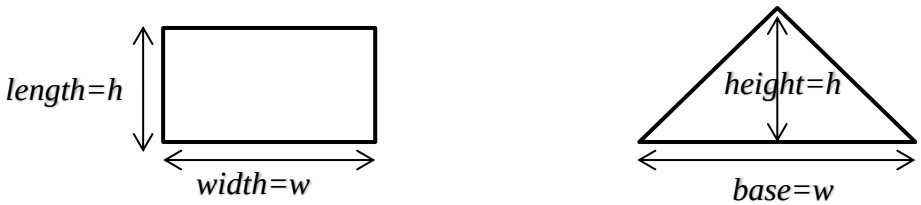
SUBMISSION PROCEDURE:

- Only the source code files are required for the submission (i.e. the edited **`program1.cpp`** and **`program2.cpp`**).
- You do not need to compress the files.
- Submit the source code files to **UTM's e-learning system**.

(This question booklet consists of 9 pages including this page.)

Program 1 **[35 Marks]**

Rectangles and triangles are two common types of polygons. They can be represented by their dimensions, *w* and *h.* As for a rectangle, *w* represents the width and *h* is the length of the rectangle. Whereas for a triangle, *w* and *h* are the base and height of the triangle, respectively (Figure 1).



(a) Rectangle (b) Triangle

Figure 1: Representation of a rectangle and triangle.

Given an incomplete program source code, **`program1.cpp`** which contains a base or parent class named **`Polygon`** and two derived or child classes named **`Rectangle`** and **`Triangle`,** respectively. Complete the program based on the tasks 1 to 5 below. Note that, the tasks are also stated in the program.

Task 1: In class **`Polygon`**, define two member functions or methods and allow them to be **polymorphic**. The methods and their purpose are as follows:

a) **`display`** : to display a message ***"Generic polygon"*** onto the screen.

b) **`calcArea`**: to return a value of zero.

(6 marks)

Task 2: Declare the class **`Rectangle`** as a child of the class **`Polygon`** and complete the definitions for the following methods:

a) the constructor: to initialize the width and length of the rectangle.

b) **`display`** : to display the width and length of the rectangle.

c) **`calcArea`**: to calculate the area of the rectangle. ***Formula:*** area = width × length.

(9 marks)

Task 3: Declare the class **`Triangle`** as another child of the class **`Polygon`** and complete the definitions for the following methods:

a) the constructor: to initialize the base and height of the triangle.

b) **`display`** : to display the base and height of the triangle.

c) **`calcArea`** : to calculate the area of the triangle. ***Formula:*** area = $\frac{1}{2}$ × base × height.

(9 marks)

Task 4: Declare an array of pointers to polygons and fill in the array with different types of polygons that are allocated dynamically. The polygons are:

- a triangle with the base is 10 and height is 20,
- a rectangle with the width is 20 and length is 20,
- a generic polygon, and
- another rectangle with the width and length are 15 and 10, respectively.

(6 marks)

Task 5: Display the information of all polygons along with their area and calculate the total area of the polygons. The program should produce screen output as shown in Figure 2.

(5 marks)

```
Polygon #1
Triangle:
        Base = 10, Height = 20
        Area = 100

Polygon #2
Rectangle:
        Width = 20, Length = 20
        Area = 400

Polygon #3
Generic polygon:
        Area = 0

Polygon #4
Rectangle:
        Width = 15, Length = 10
        Area = 150

The total area of all polygons = 650
```

Figure 2: Program output.

Program 2 **[65 Marks]**

You are given an incomplete program source code named **`program2.cpp`** for this question.

Consider the class diagram in Figure 3 which illustrates the data model for patient records in a hospital. There are two types of patients; inpatients which need to be admitted to wards (thus the information about their wards will be recorded) and outpatients which do not need to stay at the hospital. As for each outpatient, the information regarding the type of diagnosis needed will also be recorded. Regardless of either inpatients or outpatients, the hospital needs to record the patient's illness and the information about his or her guardian.

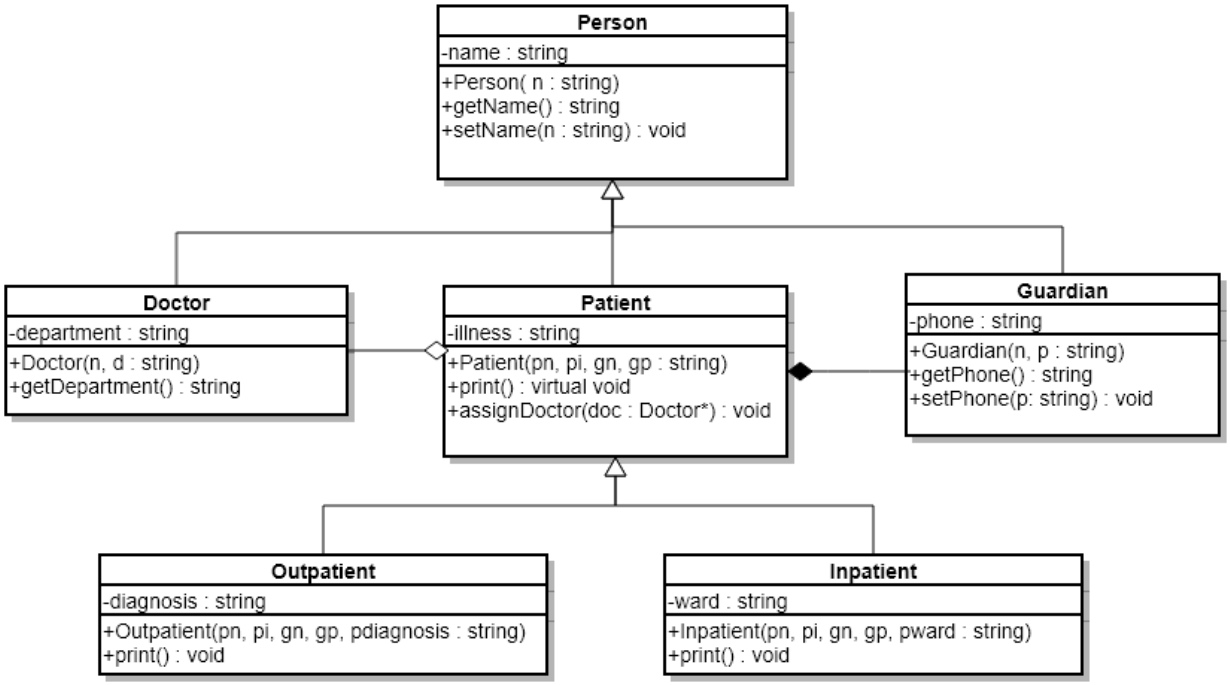


Figure 3: Class diagram for patient records.

Based on the class diagram in Figure 3, complete the given program (**`program2.cpp`**) to perform the following tasks:

a. Implement all the classes (except the class **`Person`**) with the member data or attributes and member functions or methods specified in the diagram. Note that, the definition for the class **`Person`** is fully given, whereas the definitions for the other classes are partially given. The purpose of each method is as the name implies and some of them are further explained below:

   - The method **`assignDoctor`** in the class **`Patient`** will be used to assign a doctor to a patient.

- The method **`print`** in the class **`Patient`**, **`Outpatient`**, and **`Inpatient`** will be used to print the patient's information onto the screen, accordingly.

b. Your implementation must apply some OOP principles including **aggregations**, **compositions**, **inheritances** and **polymorphisms**.

c. Create an array to hold a list of doctors and initialize the array with the following data:

Doctor's Name	**Department**
Dr. Ramli	ICU
Dr. Kamariah	Radiology

d. Create an array to hold a list of patients below. You **must use only a single array** for this.

Patient's Name	**Illness**	**Guardian's Name**	**Guardian's Phone**	**Diagnosis Needed**	**Ward Number**	**Patient Type**
Rozita	Sprained ankle	Salleh	4466	X-ray	-	Outpatient
Nurdiana	Respiratory failure	Jalil	7731	-	ICU 101	Inpatient
Ali	Coronary artery	Bakar	1234	CT Scan	-	Outpatient

e. Your program needs to provide the user a menu-driven interaction with the following options.

Menu Options	**Description**
1. List Doctors	To print all doctors onto the screen.
2. List Patients	To print all patients onto the screen.
3. Assign Doctor	To assign a patient with a doctor. The user needs to choose the respective patient and doctor using their array indices. As such, the program needs to ensure that the indices are within the valid range. Otherwise, an exception indicating out of range will be thrown. You **must use the exception handling** strategy for this purpose.
4. Exit	To end the program.

Figure 4 shows the expected result of the 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. Meanwhile, Table 1 shows the assessment criteria for this question.

Interaction 1: The user chooses option 1 to list the doctors.

```
========== Menu ==========
1. List Doctors
2. List Patients
3. Assign Doctor
4. Exit

Choose an operation [1-4] => 1

Doctor's Name: Dr. Ramli        Department: ICU
Doctor's Name: Dr. Kamariah     Department: Radiology
```

Interaction 2: The user chooses option 2 to list the patients.

```
========== Menu ==========
1. List Doctors
2. List Patients
3. Assign Doctor
4. Exit

Choose an operation [1-4] => 2

Patient #1
Patient Type:  OUTPATIENT
Diagnosis   :  X-ray

Patient's Name   :Rozita
Illness          :Sprained ankle
Guardian's Name  :Salleh
Guardian's Phone :4466
 *** No doctor assigned yet ***

Patient #2
Patient Type:  INPATIENT
Ward        :  ICU 101

Patient's Name   :Nurdiana
Illness          :Respiratory failure
Guardian's Name  :Jalil
Guardian's Phone :7731
 *** No doctor assigned yet ***

Patient #3
Patient Type:  OUTPATIENT
Diagnosis   :  CT Scan

Patient's Name   :Ali
Illness          :Coronary artery
Guardian's Name  :Bakar
Guardian's Phone :1234
 *** No doctor assigned yet ***
```

Interaction 3: The user chooses option 3 to assign doctors to patients.

```
========== Menu ==========
1. List Doctors
2. List Patients
3. Assign Doctor
4. Exit

Choose an operation [1-4] => 3

Enter the patient index and doctor index => 1 0

========== Menu ==========
1. List Doctors
2. List Patients
3. Assign Doctor
4. Exit

Choose an operation [1-4] => 3

Enter the patient index and doctor index => 2 1
```

*Interaction 4: The user chooses option 3 to assign a doctor but he or she enters **incorrect array index** for the patient or the doctor.*

```
========== Menu ==========
1. List Doctors
2. List Patients
3. Assign Doctor
4. Exit

Choose an operation [1-4] => 3

Enter the patient index and doctor index => 5 0
** Error: Index is out of range

========== Menu ==========
1. List Doctors
2. List Patients
3. Assign Doctor
4. Exit

Choose an operation [1-4] => 3

Enter the patient index and doctor index => 0 -1
** Error: Index is out of range
```

Interaction 5: The user chooses option 5 to list again the patients

```
========== Menu ==========
1. List Doctors
2. List Patients
3. Assign Doctor
4. Exit

Choose an operation [1-4] => 2

Patient #1
Patient Type:  OUTPATIENT
Diagnosis   :  X-ray

Patient's Name   :Rozita
Illness          :Sprained ankle
Guardian's Name  :Salleh
Guardian's Phone :4466
 *** No doctor assigned yet ***

Patient #2
Patient Type:  INPATIENT
Ward        :  ICU 101

Patient's Name   :Nurdiana
Illness          :Respiratory failure
Guardian's Name  :Jalil
Guardian's Phone :7731
Doctor's Name    :Dr. Ramli
Doctor's Dept.   :ICU

Patient #3
Patient Type:  OUTPATIENT
Diagnosis   :  CT Scan

Patient's Name   :Ali
Illness          :Coronary artery
Guardian's Name  :Bakar
Guardian's Phone :1234
Doctor's Name    :Dr. Kamariah
Doctor's Dept.   :Radiology
```

Figure 4: An example run of the program.

Table 1: Assessment Criteria

Item	Criteria	Marks
A	The program can run with appropriate input and output.	1
	The code is neatly written including proper use of indentations.	1
B	**Class definitions:**	
	`Doctor`	2
	`Guardian`	3
	`Patient`	7
	`Outpatient`	2
	`Inpatient`	2
C	**Implementation of OOP Principles:**	
	Aggregations	3
	Compositions	4
	Inheritances	12
	Polymorphisms	4
D	**The main program:**	
	Creating the list of doctors	3
	Creating the list of patients	5
	Listing doctors	4
	Listing patients	3
	Assigning a doctor to a patient	3
	Handling the “out of range” error using exceptions	6
	Total	**65**