Question 1 **[35 Marks]**

Instructions:

- A starter code is provided for this question in **program1.cpp.**
- For question (a) and (b), write your answers in the provided answer booklet.
- For questions (c) to (h), modify the same program**, program1.cpp.** Please adhere to the following guidelines:
  - Write in an inline style.
  - Do not modify any access specifiers in the program.
  - Do not add any additional attributes or methods to the classes.
  - Do not modify the headers of any methods or functions in the program.

Given a class diagram in Figure 1 that models a course management system.

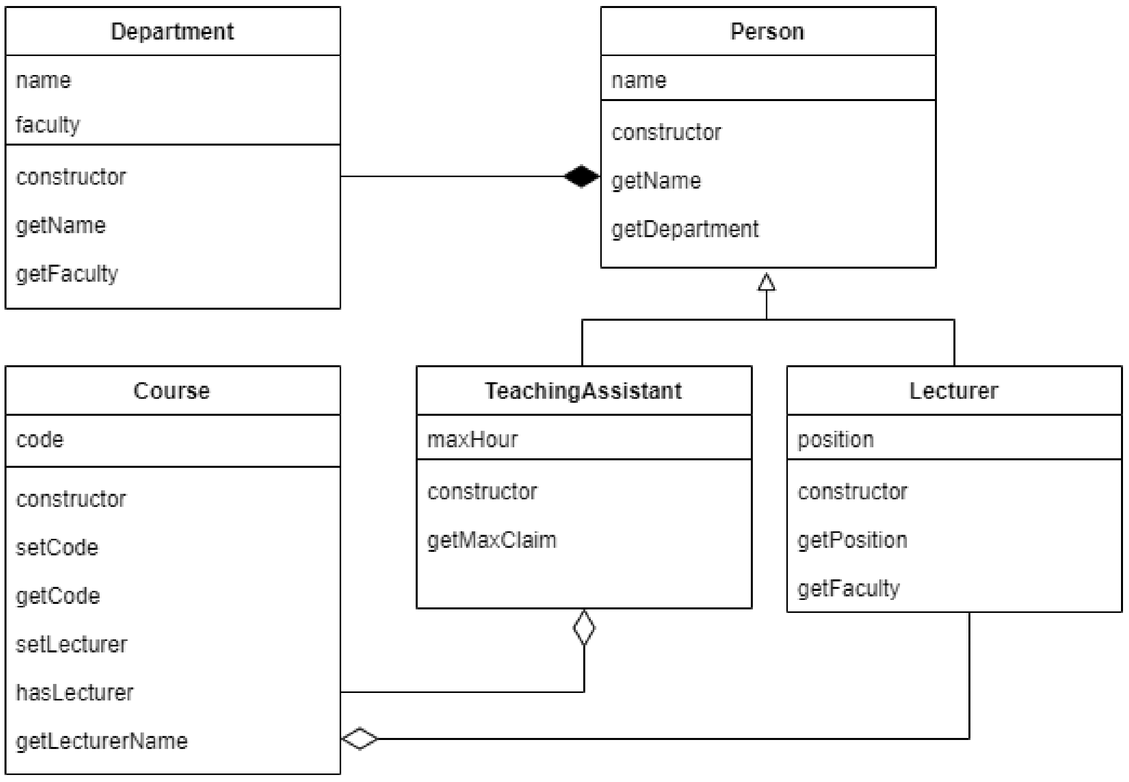

Figure 1

Analyze the diagram and answer the following questions:

a) Based on the class diagram, identify the Object-Oriented Programming relationships between the classes listed below. If no relationship exists, state "No relationship" and provide reasoning for each answer.

i) Lecturer and Department

ii) TeachingAssistant and Lecturer

iii) TeachingAssistant and Person

iv) Course and Lecturer (8 marks)

b) Distinguish the relationship between Course and TeachingAssistant from that of Course and Lecturer. (2 marks)

c) The starter code in **program1.cpp** is intended to implement the class diagram. Modify this code to accurately represent all relationships shown in the class diagram. (5 marks)

d) Complete the `Person` class in the **program1.cpp** by implementing the following methods:

i) `Person(string n="", string d="", string f="")`

A parameterized constructor that accepts several parameters, **n:** person's name, **d**: department's name and **f**: faculty's name. (1 mark)

ii) `getDepartment`

A getter method that returns a department object. (2 marks)

e) Complete the `Lecturer` class in **program1.cpp** by implementing the following methods:

i) `Lecturer(string n, string d, string f, string p)`

A parameterized constructor accepting several parameters, **n:** person's name, **d**: department's name, **f**: faculty's name and **p:** position. (1 mark)

ii) `getFaculty`

A getter method that returns the lecturer's faculty name. (2 marks)

f) Complete the `Course` class in **program1.cpp** by implementing of the following methods:

i) `Course(string c="")`

A parameterized constructor accepting a parameter, **c:** course code. (1 mark)

ii) `setLecturer`

A setter method that assigns a lecturer to a course. (1 mark)

iii) `hasLecturer`

A getter method that returns `True` if a lecturer has been assigned to the course.

(1 mark)

iv) **`getLecturerName`**

A getter method that returns the lecturer's name assigned to the course or an empty string if the course has no lecturer. (2 marks)

g) In the main function of the current version of **program1.cpp**, there are codes that create a list of courses with a regular array and code that displays the list using a regular loop approach. Rewrite the main function using the **`vector`** type. (5 marks)

h) In the current version of **program1.cpp**, there is a function named **`courseCodeToName()`** which determines the name of a course based on its code. Rewrite this function by incorporating the **`map`** type. (4 marks)

Question 2 **[65 Marks]**

Instructions:

- No starter code is provided for this question. You will have to write the program from scratch.
- As for questions (a) to (d), write your answers in the answer booklet.
- As for question (e) and (f), write a complete C++ program and save it in a single file named **program2.cpp**.

Consider the UML class diagram in **Figure 2** and answer the following questions:

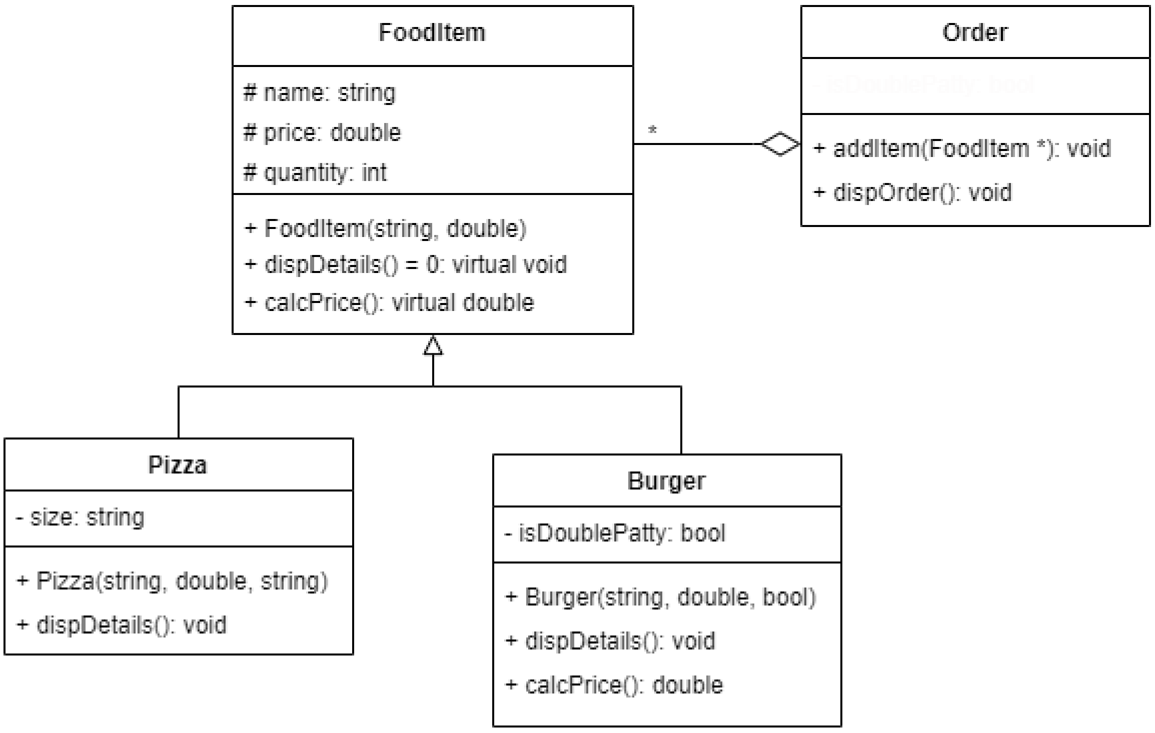

Figure 2: UML class diagram

a) Why are virtual functions used in the **`FoodItem`** class? (3 marks)

b) What are the advantages of using virtual functions compared to non-virtual functions? (3 marks)

c) What is the difference between a pure virtual function and a virtual function in C++? (6 marks)

d) When is it appropriate to use a pure virtual function in a C++ program? (3 marks)

e) Write a complete C++ program based on the UML class diagram in **Figure 2**. Implement all the classes with their member variables (attributes) and member functions (methods) as shown in the diagram. The purpose of each function is as the name implies, and more explanations are provided below. Your program should be able to generate the output shown in **Figure 3**.

i) Define the `FoodItem` class. This class should have three member functions:
   - Constructor with arguments to initialize all the member attributes of the class.
   - `dispDetails` is a pure virtual function.
   - `calcPrice` to calculate the price of the food item. It does this by multiplying the price of each food item by its quantity. Please design the function to be polymorphic.

(7.5 marks)

ii) Define the `Pizza` class as a subclass of the `FoodItem` class. This class should have two member functions:
   - Constructor with arguments to initialize all the member attributes of the class, including those inherited from the superclass.
   - `dispDetails` to display the details of a pizza, including its name, price, quantity, and size (as depicted in **Figure 3**).

(6.5 marks)

iii) Define the `Burger` class as a subclass of the `FoodItem` class. This class should have three member functions:
   - Constructor with arguments to initialize all the member attributes of the class, including those inherited from the superclass.
   - `dispDetails` to display the details of a burger, including its name, price, quantity, and whether it has a double patty or not (as depicted in **Figure 3**).
   - `calcPrice` to calculate the price of the burger. An additional charge of RM3.80 will be applied for each burger if it has a double patty.

(10 marks)

iv) Define the `Order` class. This class should have two member functions:
   - `addItem` to assign the element in the array of `Food` pointers with the passed argument. ***Note:*** For this purpose, you must use a dynamic array (`Vector`).
   - `dispOrder` to display the details of the food items in the order. Please produce the output as shown in **Figure 3**.

(9.5 marks)

v) Define the **`main`** function:

- Create an **`Order`** object.
- Create an array of **`FoodItem`** pointers that dynamically allocates **`Pizza`** and **`Burger`** objects. ***Hint:*** Use the concept of polymorphism.
- Add all the created food items in the array to the **`Order`** object. ***Note:*** You must use **`sizeof`** function to count the number of pointers in the array.
- Display the details of all the food items in the order (as depicted in **Figure 3**).

(11.5 marks)

f) Add appropriate statements to the program in (e) to handle an error if the total price of the order is below RM20.00. This should be done using an **exception handling** approach. **Figure 4** shows an example of program output when the total order price is below RM20.00.

(5 marks)

```
Pizza - Super Supreme
Price: RM52.3
Quantity: 1
Size: Large

Burger - Cheeseburger
Price: RM14.5
Quantity: 3
Double Patty: No

Pizza - Hawaiian Chicken
Price: RM19.9
Quantity: 3
Size: Personal

Pizza - Beef Pepperoni
Price: RM40.9
Quantity: 1
Size: Regular

Burger - Chicken
Price: RM13.5
Quantity: 2
Double Patty: Yes

Total Order Price: RM231
```

Figure 3: Program output

```
OrderException: The total order amount is less than RM20.
To place an online order, the minimum order amount should be RM20!
```

Figure 4: Sample output of error handling