SECTION B

- Answer **ALL** questions in the answer booklet provided.

Short Answer Questions **[15 + 15 = 30 marks]**

a) Consider the following problem statement:

You have been hired by a newly established TV/video streaming service to develop a new software application to track the titles they offer and what content is being viewed by their customers. Customers pay a monthly subscription fee to gain access to standard titles, and can optionally pay an increased fee for access to premium titles. Titles are then divided into single productions, e.g., movies, and serial productions, e.g., television series, which consist of a set of two or more single productions. The different titles offered must be searchable by title (the name of the show), genre (action, comedy mystery, etc.), and/or actor/actress. This search function must be able to be performed both by customers wishing to view titles and by the company wishing to identify which customers have viewed which titles.

Your task is to prepare a UML class diagram illustrating the concepts in the problem statement above. Your class diagram must demonstrate the following elements:

- Attributes, properties, and methods;
- Collections;
- Abstract classes;
- Inheritance or interfaces;
- Associations (any type); and
- Cardinality.

b) For this task you are required to write C# code satisfying the following requirements:

- An abstract class named Result which has:
  - An string attribute for the student name;
  - An integer attribute for the mark achieved;
  - Read-only properties encapsulating the above attributes;
  - A method ScaleUp which allows only derived classes to provide a new higher value for the mark;
  - A custom constructor which takes name and mark parameters and initialises the attributes defined above; and
  - A ToString() method which returns a string containing the asset description and value in the format:

    name: *mark*
- An class named ScaledResult which inherits from the Result class and has:
  - An integer attribute for the rate;
  - A read-only property encapsulating the above attribute;
  - A custom constructor which takes name, mark, and rate parameters, passes the name and mark to the base class constructor and initialises the rate attribute defined above;
  - A Scale() method, which can be invoked by any other class, which increases the mark of the student according to the following formula:

$$new\_mark = old\_mark + (old\_mark * \frac{rate}{100})$$

[Section B Total: 30 marks]

[Grand Total: 120 marks]