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 major online company to develop the underlying software system for a new web site which will compete with other combined auction and retail sites. The software system needs record clients who will be either individuals or companies. Both can sell items on the web site, either individual items for a once-off sale, e.g., a second-hand/used item being sold, or can be an item for repeat sales, e.g., for an individual/company selling Blu-Ray titles as part of an ongoing retailing business. Once-off items will also have a duration recorded, specifying how long the auction will remain open for (either 4, 7, or 10 days), after which time they will no longer appear on the web site. Repeat sales items will have a record of how many items are in stock and once all items are sold they will no longer appear on the web site. Individuals can also place bids for both once-off items, for which they will specify their maximum bid amount, and can purchase repeat sales items. The software system will also track how much money individuals owe for purchasing items and how much money is owed to both individuals and companies for items sold.

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 Purchase which has:
  - A string attribute for the description of the purchase;
  - A decimal attribute for the amount owed;
  - Read-only properties encapsulating the above attributes;
  - A method MakePayment which allows only derived classes to specify a payment against the amount owed, returning true if the total amount owed has been paid or false if there is still an amount owing after the payment;
  - A custom constructor which takes a parameter for description and price, initialising the attributes above (the price represents the initial amount owed); and
  - A ToString() method which returns a string containing the asset description and price in the format:

    description: *price*
- A class named CashPurchase which inherits from the Asset class and has:
  - A decimal attribute for amount paid;
  - A string attribute for change owed;
  - Read-only properties encapsulating the above attributes;
  - A custom constructor which takes description and price, passing this information to the base class constructor and initialising the above attributes as required;
  - A MakePayment() method (returns void), which accepts a decimal parameter indicating the amount paid, increases the amount paid attribute by the parameter value, and updates the change owed attribute to a string formatted as follows:

    note_value dollars in notes and *coin_value* in coin.

    For example, paying $50 for a price of $8.50 would result in a string "40 dollars in notes and 1.5 in coin" (currency format could be used to improve this, but is not required).

Hint: notes come in $5, $10, $20, $50, and $100, so the amount in notes will always be a multiple of $5, and coin value will always be less than $5.

[Section B Total: 30 marks]

[Grand Total: 120 marks]