1. (a) Explain how to merge two sorted arrays *a1* and *a2* into a third array *a3*, using diagrams to illustrate your answer. What is the time complexity of the array merge algorithm? (Note that you are *not* asked to write down the array merge algorithm itself.)

(6)

(b) Write down the array merge-sort algorithm. Use diagrams to show how this algorithm works. What is its time complexity?

(6)

(c) Develop an algorithm to merge two sorted SLLs (singly-linked lists) into a third SLL. Your algorithm should start:

> To merge the SLL headed by *first1* and the SLL headed by *first2*
> into an SLL headed by (*first3*,*last3*):

(8)

2. (a) Explain the fundamental difference between a *stack* and a *queue*. How do they both differ from a *list*?

(3)

(b) A *dequeue* (or *double-ended queue*) is a sequence of elements with the property that elements can be added, inspected, and removed at both ends. Design a dequeue abstract data type, whose elements are objects, and which enables application programs to:

(1) make a dequeue empty;

(2) add a given element at the front or rear of a dequeue;

(3) remove the element at the front or rear of a dequeue;

(4) inspect the element at the front or rear of a dequeue;

(5) test whether the dequeue is empty.

Express your design in the form of a Java interface. Each operation must be accompanied by a comment specifying the operation's observable behaviour.

(6)

(c) Show how a dequeue could be represented by a DLL (doubly-linked list), using a diagram to display the invariant of this representation.

Also draw diagrams showing the DLL representation after each step of the following sequence:

(i) make the dequeue empty;

(ii) add "ant" to the rear;

(iii) add "bat" to the front;

(iv) add "cat" to the rear;

(v) remove the front element;

(vi) remove the rear element.

(8)

(d) An alternative representation for a dequeue might be an SLL (singly-linked list) whose header contains links to both first and last nodes. Explain why the SLL representation would be inferior to the DLL representation.

(3)

3. (a) What is a *map*?

Explain briefly how a map can be represented by a BST (binary search tree).

(3)

(b) A *multimap* is a collection of (key, value) entries in which keys are not necessarily unique. An example of a multimap is one that associates countries with their official languages:

country	language
IT	Italian
DE	German
NL	Dutch
FR	French
BE	French
BE	Flemish
UK	English
IE	English
IE	Irish

Design an abstract data type, `Multimap`, representing multimaps whose keys and values are objects. Your design must enable application programs to:

(1) make a multimap empty;

(2) add a given entry to a multimap;

(3) test whether there is at least one entry with a given key in a multimap;

(4) find all the values associated with a given key in a multimap;

(5) remove all the entries with a given key from a multimap.

Express your design in the form of a Java interface. Each operation must be accompanied by a comment specifying the operation's observable behaviour.

(6)

(c) Show how a multimap could be represented by a BST. Illustrate your answer by showing how the (country, language) multimap of part (b) would be represented.

(6)

(d) Assuming your representation of part (c), explain *briefly* how each of your multimap operations would be implemented.

(5)

4. (a) Define what is meant by a *graph*. What is the difference between a *directed graph* and an *undirected graph*?

3)

(b) Explain how an airline might model its flight network by means of an undirected graph. Illustrate your answer by drawing a graph to model the network of a fictional airline Teuchtair, which has the following flights: Glasgow from/to Stornoway and Inverness; Edinburgh from/to Glasgow and Inverness; and Inverness from/to Kirkwall.

(4)

(c) Explain how the airline might model its flight network *now including information about flight times*. Each flight time consists of a departure time and an arrival time. Assume, for simplicity, that any given flight is available at the same flight time every day of the year. (You are *not* required to illustrate your answer to this part, unless you want to.)

(5)

(d) Outline how you would use such a flight network (with information about flight times) to find all possible routes from airport A to airport B. Where a route uses an intermediate airport, a connection time of at least 30 minutes must be allowed.

(8)