OLLSCOIL NA hÉIREANN
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

Sample Examination

CS2504: Algorithms and Linear Data Structures

Professor A. N. Extern
Professor J. Bowen
Dr K. T. Herley

Answer all three questions
Total marks: 80.

1.5 Hours

Question 1 [*40 marks*] *Answer all eight parts.*

(i) Suppose that M is a map (ADT Map) of Integer-String entries (Integer keys, String values) that is initially empty and to which the following sequence of operations is applied. Show the state of the map after each operation and the value returned by the operation, if any.

```
M.put(17, "seventeen");
M.put(3, "trois");
M.put(5, "funf");
M.get(3);
M.get(8);
M.put(5, "cinque");
M.get(5);
M.remove(17);
M.put(10, "deich");
```

(*5 marks*)

(ii) Give Java statements for the following ADT operations using Queue.java and ArrayBasedQueue.java as appropriate. (a) declare a variable q to refer to a queue (ADT Queue) of integer values; (b) create a queue object and make q refer to it; (c) add the integers one to five to the queue; (d) empty the queue one item at a time and print each item. (*5 marks*)

(iii) Draw a detailed sketch to illustrate how a left-justified array might be used to represent a map (ADT Map). Describe (in words) how operation get might be implemented in terms of this representation. (*5 marks*)

(iv) Draw detailed sketch of a linked list (singly-linked list) containing three nodes with elements of type Integer. If the head is declared as follows

```
LLNode<Integer> head;
```

give a Java fragment for each of the following (i) add a node/element at the beginning of the list; (ii) remove the node at the beginning of the list and print the associated element. (*5 marks*)

(v) Draw a detailed sketch of a doubly-linked representation of ADT List both for an empty list and for a non-empty list. Describe in words or diagrams as appropriate how operations get and remove might be implemented in terms of this representation. (*5 marks*)

(vi) Assume that a left-justified array S is used to represent a stack object, with the stack items occupying slots 0 to t inclusive. Give pseudocode implementations for the following ADT Stack operations: (a) size, (b) isEmpty, (c) push, (d) pop. You may ignore all error checking and array-resizing issues. (*5 marks*)

(vii) Give a pseudocode algorithm merge that takes two lists (List<Integer>) of integer values arranged in increasing order and that merges their contents into a third list (also in increasing order). (*5 marks*)

(viii) Assume that you have a merge as described above, give a pseudocode algorithm mergesort that takes a list (List<Integer>) of integer values and that rearranges the elements of the list so that they appear in increasing order. (*5 marks*)

Question 2 [*20 marks*]

ADT Priority Queue is a container abstraction. A priority queue holds *entries*: each item consists of (i) a key (also referred to as a priority) and (ii) a value. The keys can be of any "comparable" type on which an order $x \leq y$ may be defined; the values can be of any type. The ADT supports the following operations, as well as the standard size and isEmpty operations.

- insert(k, e): Insert a new entry with key k and value e into the priority queue and return the new entry. Input: K, V; Output: Entry.
- min(): Return, but do not remove, an entry in the priority queue with the smallest key. Illegal if priority queue is empty. Input: None; Output: Entry.
- removeMin(): Remove and return an entry in the priority queue with the smallest key. Illegal if priority queue is empty. Input: None; Output: Entry.

Using a representation based on the concept of a doubly-linked list, give a complete Java implementation for this ADT. State any assumptions you make about any ancillary classes you employ (LLNode *etc.*). For full marks your implementation must be able to accommodate any key type and any value type.

Question 3 [*20 marks*]

Suppose that you wished to implement a simple plagiarism analyzer to detect student Java submissions that are virtually identical.

The Java programming language has about fifty *key words*, such as "if", "while", "static" and so on that have specific meanings in the language and cannot be used as identifiers. Two programs will be regarded as *suspiciously similar* if they have the same *profile*. A program's profile is a count for each keyword of the number of occurrences of that keyword in the program, *i.e.* the number of occurrences of the keyword "if", the number of "whiles" and so on. The idea is that the usual tricks employed to disguise plagiarism, such as changing identifiers, rewording or reformatting comments, reordering methods within the program, and so on, leave the profile unaltered and so detecting programs with the same profile allows possible plagiarism cases to be flagged.

Outline a Java application that takes a list of the names of programs submitted *i.e.* List<String> (plus the name of a directory that contains the associated files) and that identifies clusters of programs that share a common profile (by printing out the names of the programs in question). It is assumed that each program bears the name of the student who submitted it and that student names are unique.

It is not expected that you provide a complete program, but your sketch, using diagrams, pseudocode or prose as appropriate, should address the main conceptual issues that a complete program would rely upon including, but not limited to, the following:

- What data structure is used to represent the profile of each individual program;
- What approach is used to determine the profile of an individual program;
- When analyzing a program, how do you distinguish between identifiers and key words.

- What data structure(s) is(are) used to hold the profiles of the various programs analyzed.
- What approach is used to identify clusters of programs with the same profile.

Ideally, a competent Java programmer ought to be able to complete the program based on your blueprint without any significant conceptual insight or ingenuity on his part.

You may assume that Java interfaces and implementations are available for all the ADTs listed on the sheet appended to this paper. State clearly any other assumptions you make about any other classes you employ.

Note: The actual exam paper will have a copy appended of the two-page sheet entitled "cs2504 ADT Summary". This sheet was distributed in lecture earlier this term and is available on the module website.