PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Midterm Examination: Semester 2 **Academic Year**: 2008-2009

Date: 21st December, 2008 **Time**: 9:00 – **11:00 (2 hours)**

Subject Number: 240-421 **Room**: A201

Subject Title: Advanced Data Structures and Algorithms

Lecturer: Aj. Andrew Davison

Exam Duration: 2 hours

This paper has 4 pages.

Authorized Materials:

- Writing instruments (e.g. pens, pencils).
- Books (e.g. dictionaries) and calculators are **not** permitted.

Instructions to Students:

- *Answer questions in English.* Perfect English is **not** required.
- Attempt all questions.
- Write your answers in an answer book.
- Start your answer to each question on a new page
- Clearly number your answers.
- Any unreadable parts will be considered wrong.
- When writing programs, use good layout, and short comments; marks will not be deducted for minor syntax errors.
- The marks for each part of a question are given in brackets (...).

Question 1 (30 marks; 30 minutes)

a) Implement the sequential search method:

```
public static int seqSearch(int[] arr, int first,
                                    int last, int target);
```

seqSearch() searches for target in arr[] starting from the first index position, and stopping after examining position last-1. It returns the index position of target, or -1 if target is not found. (5)

b) Implement the method:

```
public static boolean isUnique(int[] arr, int target);
```

isUnique returns true if target appears only once in arr[]; false otherwise. Use seqSearch() from part (a) to implement isUnique(). (10)

c) Implement the method:

```
public static int numUnique(int[] arr);
```

numUnique() returns the number of unique values in the array. Use isUnique() from part (b) to implement numUnique(). (5)

d) Work out the worst case running time for numUnique(), showing **all** your working. (10)

Question 2 (30 marks; 30 minutes)

a) Draw a diagram showing how mergesort() splits and then merges the following array when sorting it:

```
Integer[] arr = {25, 16, 7, 19, 3, 48};
```

Do **not** include any mergesort() code. (5)

b) Draw a diagram showing how quicksort() splits and then combines the following array when sorting it:

```
Integer[] arr = {25, 16, 7, 19, 3, 48};
```

Assume that the pivot is the midpoint position of the current sub-range (calculated using integer division).

Do **not** include any quicksort() code. (5)

c) Compare mergesort() and quicksort() in terms of their worst case running times **and** space requirements. Explain the comparisons in words. (5)

d) A sorting algorithm is *stable* if two items with the same value are not rearranged with respect to each other during the sorting. For instance, in the five-element array:

$55\ 5_1\ 12\ 5_2\ 33$

A stable sorting algorithm means that the final ordering is:

$5_1\ 5_2\ 12\ 33\ 55$

The first 5 (5_1) and the second 5 (5_2) remain in the same order with respect to each other.

Explain in words, with brief examples, if mergesort() and quicksort() are stable algorithms. (10)

e) Give an example of when a stable sorting algorithm is important. Do **not** include any code. (5)

Question 3

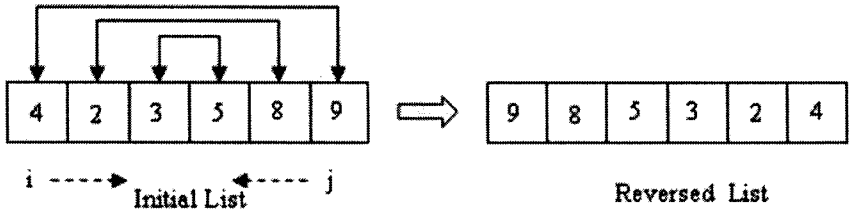
(20 marks; 20 minutes)

In this question, assume the existence of an ArrayList class; you do **not** have to implement the ArrayList class.

a) Implement the method:

```
public static <T> void reverseByIndex(ArrayList<T> aList);
```

reverseByIndex() uses indicies to reverse the order of the elements in an ArrayList, as shown in the diagram below. (5)



b) Implement the method:

```
public static <T> ArrayList<T> reverseByCopy(ArrayList<T> aList);
```

reverseByCopy() copies the elements of an ArrayList in reverse order to a *new* ArrayList, which becomes the return value of the method. (5)

c) Compare the worst case running times **and** space requirements of the methods in part (a) and (b). Explain the comparisons in words. (10)

Question 4 (40 marks; 40 minutes)

In this question, assume the existence of the Node<T> class described in the Linked Lists notes. You do **not** have to implement the Node<T> class.

a) Draw a diagram showing the list of Nodes created by the following code: (5)

```
Node<Character> p = new Node<Character>('X');
Node<Character> q = new Node<Character>('A');
Node<Character> r = new Node<Character>('M');
q.next = r;
r.next = p;
```

b) Following on from part (a), draw a diagram showing the list of Nodes created by the code: (5)

```
Node<Character> newNode = new Node<Character>('T');
Node<Character> nextNode = q.next;
q.next = nextNode.next;
```

c) What is the running time for an insertion at the *front* of an n-element singly linked list? Explain your answer in words, and with a diagram. (5)

d) What is the running time for an insertion at the *back* of an n-element singly linked list? Explain your answer in words, and with a diagram. (5)

e) Implement the method:

```
public static <T extends Comparable<? Super T>>
       Node<T> insertOrder(Node<T> front, T item);
```

insertOrder() inserts an item into a singly linked list *which stores its nodes in ascending order*. The front of the list is the return value. (20)

--- End of Examination ---