PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Midterm Examination: Semester 2 **Academic Year**: 2009-2010

Date: 24th December, 2009 **Time**: 13:30 – **15:30 (2 hours)**

Subject Number: 241-423 **Room**: A401

Subject Title: Advanced Data Structures and Algorithms

Lecturer: Aj. Andrew Davison

Exam Duration: 2 hours

This paper has 3 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) What are the advantages of using *generics* rather than Object references for general-purpose classes and methods? Explain in words, with the help of diagrams and small code fragments. (10)

b) A generic binary search method takes an array `arr`, indicies `low` and `high` that describe the index range, and a `target` value. It scans the array looking for a match for `target`, and returns the match's index, or -1 if no match is found.

Implement a *recursive* version of the binary search algorithm using a divide and conquer strategy. (20)

Question 2

(60 marks; 60 minutes)

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

```
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:

```
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. (10)

d) You must sort an array of n integers $\{a_0, a_1, ..., a_n\}$, each of which is between 0 and m-1 for a fixed m value. For example, sort {49, 26, 17, 0, 9, 17, 8}, which are between 0 and 49, so m is 50.

A very fast sort algorithm for solving this type of problem is called a *bucket sort*. An array of m counters, or *buckets*, is used. Each of the counters is set to zero initially.

All of the array is examined, using the counters to store totals of the number of occurrences of each value between 0 and m-1. For the example above, there would be 50 counters. The counter for the number of 0's would be set to 1, the counter for the number of 17's would be set to 2, and so on.

The sorted result is produced by examining the counters:

- The counter for the number of 0's is used to place that number of 0's in the array.

- The counter for the number of 1's is used to place that number of 1's in the array.
- And so on, up to the counter for m-1.

Write a Java static function, called bucketSort(), which sorts an array of integers, using m counters. (20)

e) Calculate the worst case running time for bucketSort(). Define the expression in terms of the size of the array, n, and the number of counters, m. (10)

f) The worst case analysis for bucketSort() is very inaccurate. Informally explain why the running time is actually *linear*: O(m + n). (10)

Question 3

(30 marks; 30 minutes)

a) For what type of operations is a singly linked list more efficient than an array. Explain in words, with the help of diagrams. No coding is necessary. (5)

b) For what type of operations is an array more efficient than a singly linked list. Explain in words, with the help of diagrams. No coding is necessary. (5)

c) A *deque*, or doubly-ended queue, is a data structure that allows insertions and deletions at the front and back of a list, but nowhere else.

A deque can be implemented using inheritance *or* by using a private list as an internal data structure. Explain in words which implementation is better. (10)

d) Implement the deque's constructor, addFront(), and removeLast() methods using the approach you chose in part (c). Note: do **not** implement the ArrayList or LinkedList classes – assume they already exist. (5)

e) In your deque, what is the worst case running times for the addFront() and removeLast() operations. Explain your answer in words, with the help of diagrams. No coding is necessary. (5)

--- End of Examination ---