PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Final Examination: Semester 2 **Academic Year**: 2013-2014

Date: 28th February, 2014 **Time**: 9:00 – **12:00 (3 hours)**

Subject Number: 241-423 **Room**: S101

Subject Title: Advanced Data Structures and Algorithms

Lecturer: Aj. Andrew Davison

Exam Duration: 3 hours

This paper has 5 pages.

Authorized Materials:

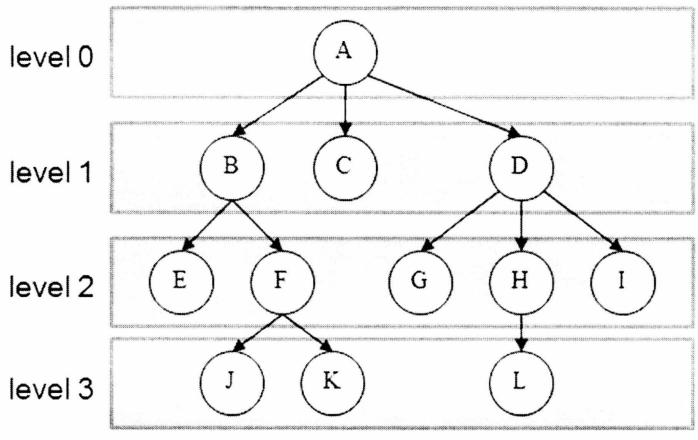
- Writing instruments (e.g. pens, pencils).
- Books (e.g. dictionaries) and calculators are **not** permitted.

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)

Consider the following tree. Each node has an associated level.



Each tree node is represented by an object that implements the Tree<T> interface:

```
public interface Tree<T>
{
  boolean isEmpty();
  T getNode();
  int getDegree();   // the number of children
  Tree<T> getSubtree(int i);
}
```

A *breadth-first traversal* visits all the nodes of a tree in the order of their levels. At each level, the nodes are visited from left to right. The tree traversal in the diagram visits the nodes from A to L in alphabetical order.

a) Implement a breadth-first traversal of a tree by using a queue. The static function should be called breadthFirstTraversal(). To begin the traversal, the root node of the tree is enqueued. Then, repeat the following steps until the queue is empty:

- Dequeue and visit the first node in the queue.
- Enqueue its children in order from left to right. (20)

Do not implement the Queue collection of the Tree<T> interface; assume they already exist.

b) What is the big-oh running time of breadthFirstTraversal() when visiting a tree of n nodes? Explain your answer. (10)

Question 2 (45 marks; 45 minutes)

A *multiset* (or bag) is a set whose elements may appear more than once. Its interface is:.

```
public interface MultiSet<T>
{
  boolean isEmpty();
  boolean add(T elem);
  boolean remove(T elem);
  int size();
}
```

a) Implement the multiset interface using a list of *unordered* pairs of the form $\{i, n_i\}$. i is an element and n_i a count of the number of i's in the multiset. This will require you to implement a Pair<T> class. (30)

b) Write a mathematical expression for the total memory space required to represent a multiset containing n instances of m distinct elements. (5)

c) What are the advantages and disadvantages of using an *ordered* list for multiset implementation? [Do not include any code in your answer.] (10)

Question 3 (40 marks; 40 minutes)

a) Heapify the following array: {14, 19, 12, 9, 22, 32, 8}, creating a max heap. Draw all the stages in the heapification, and summarize what is happening. Also show the resulting array. [Do not include any code.] (10)

b) Perform a Heap sort on the array result from part (a). *Only draw the first three stages* in the sort, where the largest three elements of the array are put into ascending order position. Explain in words what is happening. [Do not include any code.] (20)

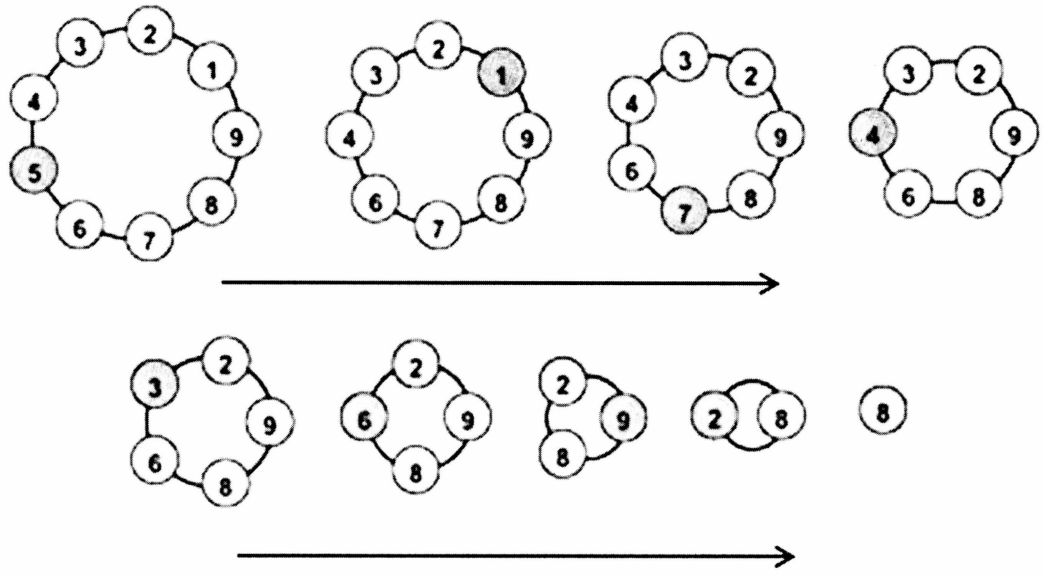
c) Draw the final sorted heap, and the corresponding sorted array. Briefly explain in words the number and types of operations that led to this result. [Do not include any code.] (10)

Question 4 (35 marks; 35 minutes)

N people have decided to elect a leader by arranging themselves in a circle and eliminating every M th person around the circle, closing ranks (i.e. the circle shrinks) as each person is 'deleted' . Each person is assigned a unique ID (1 to M), and counting starts at ID '1', and moves around the circle in ascending ID order. Counting restarts at the next person after someone is removed.

The problem is to find out which person will be the last one remaining. This function of N and M is called the *Josephus function*.

We also want to know the order in which the people are 'deleted'. For example, as shown in the figure if N = 9 and M = 5, people are removed in the order 5 1 7 4 3 6 9 2, and 8 is the ID of the chosen leader.



a) Write a function to execute the Josephus function for general values of M and N. Use the Node<T> data structure from the notes to build a circular "pointer-based" data structure. The function should print the IDs of the people being removed, and return the ID of the chosen leader. (20)

b) The figure above implements the Josephus function as a circular linked list. However it is possible to represent the problem as **two** arrays using indices instead of references for links.

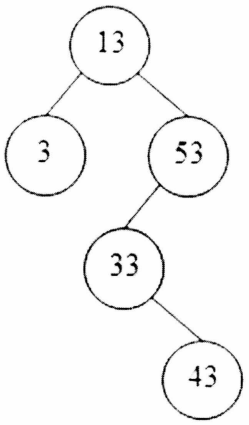
Draw a diagram showing the first four stages of the Josephus function (where IDs 5 1 7 4 are removed) when encoded with arrays. Explain in words how the arrays are being used. [Do not include any code.] (15)

Question 5 on the next page.

Question 5

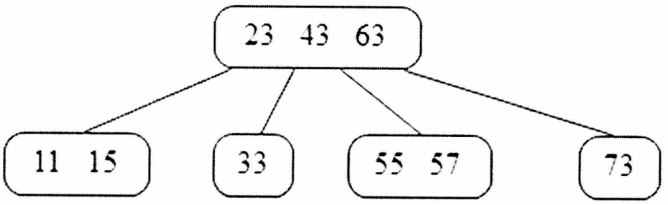
(30 marks; 30 minutes)

a) Carry out rotations on the following tree to create an AVL tree. (10)



Draw a diagram for each rotation of the tree, and explain each rotation in words.

b) Create a red-black tree for the following 2-3-4 tree. (10)



Draw a diagram for each transformation of the tree, and explain each transformation in words.

Make sure you clearly indicate (via words or color) which of the nodes of the resulting red-black tree are RED, and which are BLACK.

c) Explain the necessary properties of a red-black tree. (10)

--- End of Examination ---