PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Midterm Examination: Semester 2

Academic Year: 2011-2012

Date: 24th December, 2011

Time: 13:30 – **15:30 (2 hours)**

Subject Number: 241-423

Room: S817

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:

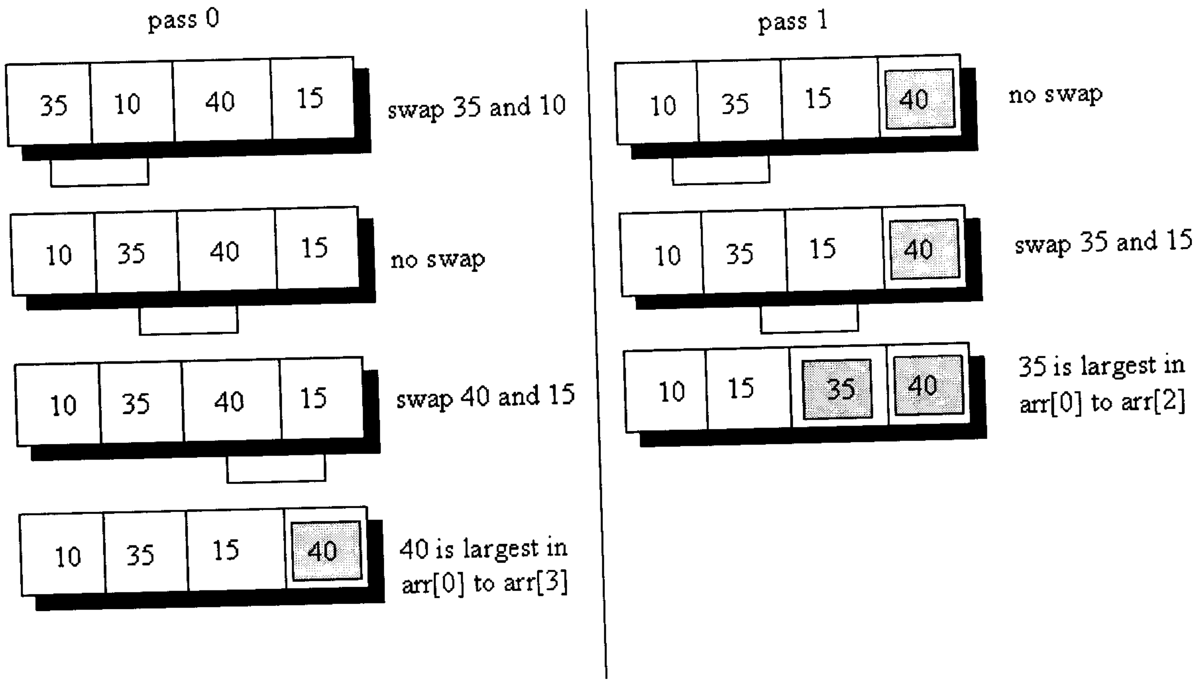
- *Answer questions in English.* Perfect English is **not** required.
- Attempt all questions.
- Write your answers in an answer book.
- Start your answer to each question on a new page
- Clearly number your answers.
- Any unreadable parts will be considered wrong.
- When writing programs, use good layout, 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: The Bubble Sort (30 marks; 30 minutes)

On each pass through a bubble sort, adjacent array elements are compared, and exchanged when the left element is greater than the right. A boolean flag is set to true if a swap occurs. At the end of a pass, the largest element has "bubbled up" to the top of the array.

The sort requires at most n-1 passes with the algorithm terminating if the flag stays false (i.e. there have been no swaps during the pass).

The following illustrates two passes of the bubble sort for the array {35,10,40,15}. The boxed elements are in their correct location.



a) Implement the bubbleSort() method: (15)

```
public static void bubbleSort(Object[] arr)
{ . . . }
```

b) Calculate the Big-Oh for the worst-case running time. (5)

c) Explain in words, without calculation, the **best-case** and **average case** Big-Oh values for bubble sort. (5)

d) Computer scientist, *Donald Knuth*, considers that "the bubble sort seems to have nothing to recommend it, except a catchy name". Can you suggest a relatively simple way to speed up bubble sort? Explain in words and diagrams, do **not** code up your suggestion. (5)

Question 2

(30 marks; 30 minutes)

Binary search takes an array arr, indicies first and last that describe an index range, and a target value, and scans the list looking for a match. The method returns the index of the match, or -1 if no match occurs.

a) Implement a **recursive** version of the algorithm using a divide-and conquer strategy. (15)

```
public static <T extends Comparable<? super T>
       int binSearch(T[] arr, int first, int last, T target)
{...}
```

b) Calculate the Big-Oh for the worst-case running time. (15)

Question 3

(30 marks; 30 minutes)

OrderedBag is a generic collection class that extends the Bag class, whose method signatures and protected data are listed below.

```
public class Bag<T> implements Collection<T>
{
  protected T[] bagArr;       // data storage
  protected int bagSize;      // no. of elems in the bag

  public Bag(int capacity);
  public boolean add(T item);
  public boolean contains(Object item);
  public boolean remove(Object item);
  public int size();
  public void clear();
  public boolean isEmpty();
  public String toString();
  public T grab();
}
```

OrderedBag overrides the **add()** method by inserting a new item in ascending order. The class also defines the methods **getFirst()** and **getLast()** which take advantage of the ordering. They return the value of the minimum and maximum element in the collection respectively.

a) Implement the OrderedBag class. Do **not** implement the methods in the Bag class. (20)

b) Calculate the Big-Oh worse case running time for add(), and also discuss the average running time. (10)

Question 4

(30 marks; 30 minutes)

Assume the existence of a DNode class for implementing a doubly linked list:

```
public class DNode<T>
{
  public T nodeValue;
  public DNode<T> prev, next;


  public DNode(T item)
  {
    nodeValue = item;
    next = this;
    prev = this;
  }
}
```

a) Explain, with the help of short examples and diagrams, how the DNode class could be used to build a **circular doubly linked list** of three Integer values. The list must be implemented using a **sentinel node**. (20)

   A sentinel node (or header) is a DNode object containing a null data value, which the doubly linked list never changes. The actual data items in the list begin with the successor of the header node. The last data item in the list is the predecessor of the header node.

b) Implement the contains() method which tests if a given item is contained in the list: (5)

```
public static <T> boolean contains(DNode<T> header, T item)
{ . . . }
```

c) Explain the advantages of using a sentinel node in a circular list by comparing it with another approach to implementing the list. (5)

--- End of Examination ---