Aston University

School of Engineering and Applied Science

CS2310 Data Structures and Algorithms with Java

Second Year Examination

CLOSED BOOK

Date: 15th January 2013
Time: 09:30 – 11:00
Duration: 1 hour 30 minutes

Instructions to Candidates

1. Answer ALL questions from Section A (50 marks)
2. Section A contains SIX questions
3. Answer TWO questions from Section B (50 marks)
4. Section B contains THREE questions, each question is worth 25 marks.
5. Use of calculators is NOT allowed

Materials provided

1. Answer booklets

This exam paper cannot be removed from the exam room

This examination is subject to the Examination Regulations for Candidates REG/08/415(2)

In all questions that involve the writing of Java program code, a few MINOR syntactic errors will not be penalised. Program code that contains more substantial errors may still gain a substantial proportion of the available marks provided the intended meaning of the code is clear. Thus candidates are advised to include appropriate comments that briefly explain the intended meaning of their code.

Section A — Answer ALL questions in this section

1. Consider the following phrases:

   - executes in quadratic time
   - executes in linear time
   - executes in log-linear time
   - executes in constant time
   - executes in logarithmic time
   - executes in exponential time
   - executes in cubic time

   a) Making use of the above phrases, state the meaning of EACH of the following expressions in the **Big-O** notation:

      i) executes in $\mathrm{O}(1)$ time

      ii) executes in $\mathrm{O}(\mathrm{n}\ \log\ \mathrm{n})$ time

      iii) executes in $\mathrm{O}(3^{\mathrm{n}})$ time

   (3 marks)

   b) Making use of the above phrases, state the time complexity of EACH of the following **growth functions**:

      i) $T(n) = 65n$

      ii) $T(n) = 5n^2 + \frac{2^n}{4500} + 20$

   (2 marks)

2. Consider the following Java code for calculating the n^{th} **Fibonacci number**:

```
1 public static int Fib(int n) {
2         if (n ==0 || n== 1) {
3                 return 1;
4         } else {
5                 return Fib(n-1) + Fib(n-2);
6         }
7 }
```

Making use of a suitable **Big-O** notation, explain in some detail the time complexity of method `Fib`.

(4 marks)

3. a) **Arrays**, **linear linked structures** and **hash tables** are three different data structures. Making use of suitable diagrams, briefly describe the key differences in their structures.

(9 marks)

b) Briefly explain how the underlying structures of **arrays**, **linked structures** and **hash tables** influence the efficiency in the retrieval of data from EACH of these data structures.

(6 marks)

4. a) Illustrate the **insertion sort** algorithm for sorting the following names in alphabetical order:

Ali Betty Katy Felix Ben

Your answer must include ALL *steps in sorting the given sequence of names.*

(6 marks)

b) State the time complexity of the **insertion sort** algorithm. (2 marks)

c) Name ONE sorting algorithm that is known to be more time efficient than **insertion sort**.

(2 marks)

5. Consider the following definition of class `LinkedStack`. Class `LinkedStack` models some basic operations of an **unbounded stack** using a linear linked structure and contains a static inner class named `Node`.

```
1  public class LinkedStack<T> {
2
3    private Node<T> top; // the top the stack
4    private int count;   // the number of elements in the stack
5
6    public LinkedStack() {
7      top = null;
8      count = 0;
9    }
10
11   public void push(T element) {
12     // definition omitted
13   }
14
15   public T pop() {
16     // definition omitted
17   }
18
19   public boolean isEmpty() {
20      return (top == null);
21   }
22
23   // Other method definitions omitted
24
25   private static class Node<E> {
26     public E element;
27     public Node<E> next;
28
29     public Node(E element) {
30       this.element = element;
31       this.next = null;
32     }
33   } // END OF CLASS Node
34
35 } // END OF CLASS LinkedStack
```

a) Write Java code for implementing method `push` in class `LinkedStack`.

(4 marks)

b) Write Java code for implementing method `pop` in class `LinkedStack`.

(6 marks)

6. Consider the following definitions for classes `Entry` and `Meaning`. Class `Entry` uses class `Meaning` to model a word entry in an electronic dictionary.

```
1  import java.util.Set;
2  public class Entry {
3     private String word;
4     private Set<Meaning> meanings; // one word can have >=1 meanings
5
6     public String getWord() {
7        return word;
8     }
9
10    public Set<Meaning> getMeanings() {
11       return meanings;
12    }
13
14    // constructor and other method definitions omitted
15 }
```

```
1  public class Meaning {
2     // the syntactic category, e.g. noun, verb, adverb, etc
3     private String category;
4     private String description; // a description of the word meaning
5
6     // constructor and other method definitions omitted
7  }
```

Suppose the electronic dictionary application is to use an **ordered list** to store the collection of `Entry` objects. Write out the necessary modifications to class `Entry` so as to enable the entries in the electronic dictionary application to be ordered alphabetically.

You should use the given line numbers to clearly indicate the location(s) of your modifications.

(6 marks)

Section B — Answer TWO questions in this section

Each question in this section carries 25 marks.

7. a) Making reference to the definition of the ADT **queue**, explain what is meant by a **priority queue**.

(5 marks)

b) Making use of suitable diagrams, describe TWO distinct ways of implementing a **priority queue**.

(8 marks)

c) Compare the time complexities of the operations for adding and removing an element from a priority queue in the two implementations described in part (b).

(4 marks)

d) Consider the following partial definition of class `CircularArrayQueue`. A `CircularArrayQueue` object models the behaviour of a standard queue using the concept of **circular array**.

```
1  public class CircularArrayQueue<T> implements QueueADT<T> {
2
3     private T[] contents;
4     private int front;  // index of first element
5     private int rear;   // index of next available slot
6
7     /** Constructor */
8     public CircularArrayQueue(int capacity) {
9        contents = (T[]) (new Object[capacity]);
10       front = 0;
11       rear = 0;
12    }
13
14    /** Returns the number of elements in the queue */
15    public int size() {
16          // implementation detail omitted
17    }
18
19    // other methods omitted
20 }
```

Write Java code to implement the above method `size()` which executes in **constant time** regardless of the number of elements in the queue.
(Hint: Whenever an element is dequeued from a `CircularArrayQueue` *object, the respective cell in array* `contents` *will be set to* `null`*.)*

(8 marks)

8. a) Given a **binary tree** structure, state what is meant by:

   i) pre-order traversal

   ii) in-order traversal

   iii) post-order traversal

   (6 marks)

   b) Consider the following partial implementation of a generic **binary search tree** class:

```
1  public class BST<T> {
2      private BinaryTreeNode<T> root; // the root node of this tree
3
4      /** Constructor: Creates an empty tree. */
5      public BST() {
6          root = null;
7      }
8
9      // inner class
10     protected static class BinaryTreeNode<E> {
11         // reference to the left sub-tree
12         public BinaryTreeNode<E> left;
13         // reference to the right sub-tree
14         public BinaryTreeNode<E> right;
15         public E element;          // the element stored in this node
16
17         // Creates a leaf node storing the specified element.
18         public BinaryTreeNode(E elem) {
19             left = null;
20             right = null;
21             element = elem;
22         }
23     }  // End of inner class
24
25     // Other methods omitted
26 }
```

   i) The header of class `BST` contains an error. Correct this error.

   (2 marks)

   ii) Making use of **recursion**, write a `public` method (plus any necessary helper methods) that returns the **size** of a binary search tree.

   (6 marks)

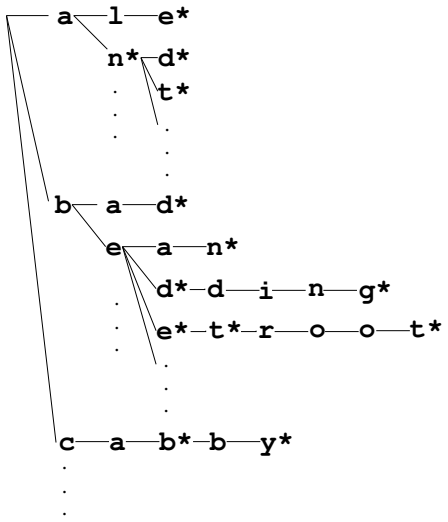
   iii) Write a `public` method for the inner class `BinaryTreeNode` that determines whether the `BinaryTreeNode` object represents a **leaf** node of a binary search tree.

   (3 marks)

(question continues on next page. . .)

(Question 8 continued. . .)

c) Consider the following **trie**:



i) Making use of a suitable example, briefly describe ONE key difference between a **trie** and a **tree** data structure.

(3 marks)

ii) Making reference to the above **trie**, briefly explain why the **trie** data structure is more suitable for facilitating the typical process of a dictionary application than a **binary search tree**.

(5 marks)

9. a) Write Java code to define a generic Java interface called `SetADT`. This interface should specify TEN typical operations of a **set**. Instances of classes which implement `SetADT` must also support the operation of an **enhanced** `for` statement.

(12 marks)

b) Name THREE concrete implementations of **set** in the Java Collections Framework (JCF).

(3 marks)

c) Which concrete implementation of **set** in the JCF is most suitable for modelling records in a University league table? Explain your answer.

(5 marks)

d) Briefly describe TWO similarities and THREE differences between the ADTs **set** and **multiset**.

(5 marks)

END OF EXAMINATION PAPER