Aston University

School of Engineering and Applied Science

CS2310 - Data Structures and Algorithms with Java

Second Year Examination

CLOSED BOOK

Date: 19th January, 2015
Time: 14:00 – 15:30
Duration: 1 hour 30 minutes

Instructions to Candidates

1. **Answer ALL questions from Section A (50 marks)**
2. **Section A contains SEVEN 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

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 $\mathrm{O(n \log n)}$ time

      ii) executes in $\mathrm{O(4^n)}$ time

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

   (3 marks)

   b) Making use of an appropriate **Big-O** notation, state the time complexity of the algorithms described by EACH of the following **growth functions**:

      i) $T(n) = 4500$

      ii) $T(n) = 5n + 20 + \frac{2}{3}nlog_2n$

      iii) $T(n) = 35^{1000} + \frac{1}{800}n + 28n^2$

   (3 marks)

2. Consider the following Java code for determining whether there are at least two values in the specified Boolean array that are **`true`**:

```
1  public static boolean has2TrueValues(boolean[] array)
2  {
3      boolean result = false;
4
5      for (int i = 0; i < array.length; i++) {
6          for (int j = 0; j < array.length; j++) {
7              if (array[i] && array[j] && (i != j)) {
8                  result = true;
9              }
10         }
11     }
12
13     return result;
14 }
```

Making use of a suitable **Big-O** notation, explain in some detail the time complexity of method `has2TrueValues`.

Your explanation should include a justification for the chosen **Big-O** *notation and make reference to the number of combination of values being checked at Line 7.*

(5 marks)

3. With the aid of a suitable diagram, briefly describe what is meant by a **stack** data structure.

(5 marks)

4. a) Making reference to the way in which data is stored within a hash table, explain what is meant by **collision** in a **hash table**.

(4 marks)

b) **Chaining** is a way to resolve collision in a hash table. Making use of suitable diagrams, outline TWO implementations of chaining.

(6 marks)

5. a) Describe what is meant by the Abstract Data Type (ADT) **set**.

(6 marks)

b) Name ONE concrete implementation of a **set** in the Java Collections Framework (JCF).

(1 mark)

6. Consider a standalone, single-user Java application which acts as a basic electronic English-to-French dictionary. In this dictionary, each English word is simply associated with its French equivalents without differentiating the part-of-speech (eg noun, verb, adjective) of the English word. This Java application loads all data into the computer's main memory when it starts up.

A user of this dictionary application:

- TYPICALLY searches for the French equivalents of an English word, and
- OCCASIONALLY prints the entire dictionary on paper, with the dictionary entries listed alphabetically.

Which type(s) of collection objects defined in the Java Collections Framework (JCF) is/are suitable for modelling the dictionary data in this application so as to support the above operations efficiently? How would the dictionary entries be modelled by your identified type(s) of collection? Explain your answer in some detail.

(10 marks)

7. Suppose you need to use **binary search** to determine whether a given element is in a collection containing 8 elements. Using a suitable diagram, illustrate the **binary search** algorithm.

Your diagram must show the size of the **search pool** *at* EACH *step of the search process.*

(7 marks)

Section B — Answer TWO questions in this section

Each question in this section carries 25 marks.

8. Consider the following definition of class `LinkedQueue`. Class `LinkedQueue` models some basic operations of an **unbounded queue** using a linear linked structure and it contains a static nested class named `LinearNode`.

```java
public class LinkedQueue<E> {

  private LinearNode<E> first; // the front the queue
  private LinearNode<E> last; // the end the queue

  public LinkedQueue() {
    first = null;
    last = null;
  }

  public void enqueue(E element) {
    // definition omitted
  }

  public E dequeue() {
    // definition omitted
  }

  public boolean isEmpty() {
     return (first == null);
  }

  // Other method definitions omitted

  private static class LinearNode<T> {
    public T element;
    public LinearNode<T> next;

    public LinearNode(T element) {
      this.element = element;
      this.next = null;
    }
  } // END OF CLASS LinearNode

} // END OF CLASS LinkedQueue
```

(question continues on next page. . .)

(Question 8 continued. . .)

a) Write Java code for method `enqueue` in class `LinkedQueue`. (6 marks)

b) Write Java code for method `dequeue` in class `LinkedQueue`. (6 marks)

c) **Array**, **linked structure** and **hash table** are three types of data structures that can be used to implement an ADT. Which data structure is best-suited for implementing an **unbounded queue**? (2 marks)

d) Making reference to the characteristics of **array**, **linked structure** and **hash table** and with the aid of suitable diagrams, explain your choice of data structure in part (c). (9 marks)

e) Making use of the classes and interfaces defined in the Java Collections Framework (JCF) ONLY, write a Java statement to define and initialise a local variable with an **unbounded queue** object for storing `Car` objects. Upon creation, the unbounded queue is expected to be empty. (2 marks)

9. a) Making use of an example, briefly describe what is meant by an **ordered list**.

(4 marks)

b) Write Java code to define a generic Java interface called `OrderedListADT`. This interface should specify NINE typical operations of an **ordered list**. Instances of classes which implement `OrderedListADT` must also support the operation of an **enhanced `for`** statement. (12 marks)

c) Name TWO concrete implementations of an **indexed list** in the Java Collections Framework (JCF). (2 marks)

(question continues on next page. . .)

(Question 9 continued. . .)

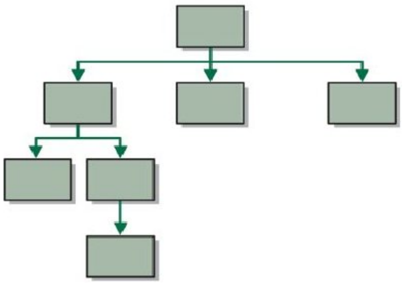
d) The following class `Dish` models a dish in a restaurant menu.

Indicate clearly what changes and additions need to be made to class `Dish` in order that instances of the class may be stored in an **ordered list**. Items in the list are to be ordered in ascending order of the fields `type` and then `price`. The same type of dishes with the same price are to be stored in alphabetical order of the field `name`.

```java
public class Dish {

    private int type;
    private int price;
    private String name;

    public Dish(int type, int price, String name) {
      this.type = type;
      this.price = price;
      this.name = name;
    }

    public int type() {
      return type;
    }

    public int price() {
      return price;
    }

    public void setPrice(int newPrice) {
      price = newPrice;
    }

    public String name() {
      return name;
    }
}
```

(7 marks)

10. a) Consider the following **tree**:



i) What is the **size** of the tree? (1 mark)

ii) What is the **height** of the tree? (1 mark)

iii) What is the **order** of the tree? (1 mark)

b) Consider the following incomplete definition of class `TernaryTreeNode`. This class models a node in a **tree** similar to the above.

```
public class TernaryTreeNode<E> {
      // Fields

      /** Constructor to create a TernaryTreeNode */

      // method definitions
}
```

i) Write Java code to define the fields for class `TernaryTreeNode`.

(4 marks)

ii) Write Java code for method `isLeaf()` in class `TernaryTreeNode`. This method determines whether the `TernaryTreeNode` object models a **leaf** node of a **ternary tree**.

(3 marks)

(question continues on next page. . .)

(Question 10 continued. . .)

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

```
public class BinaryTree<T> {
    // the root node of this tree
    private BinaryTreeNode<T> root;

    /** Constructor: Creates an empty tree. */
    public BinaryTree() {
        root = null;
    }

    // static nested class
    protected static class BinaryTreeNode<E> {
        public BinaryTreeNode<E> left;
        public BinaryTreeNode<E> right;
        public E element;

        public BinaryTreeNode(E elem) {
            left = null;
            right = null;
            element = elem;
        }
    }  // End of static nested class

    // Other methods omitted
}
```

Making use of **recursion**, write a `public` method (plus any necessary helper methods) that returns the **size** of a binary tree. (6 marks)

d) Making use of a suitable diagram, explain the main difference between a **binary tree** and a **binary search tree**. (5 marks)

e) Briefly describe TWO common uses of the **tree** data structure. (4 marks)

END OF EXAMINATION PAPER