Aston University

School of Engineering and Applied Science

CS2310 – Data Structures and Algorithms with Java

Second Year Examination

CLOSED BOOK

Date: 17th January 2018
Time: 14:00 – 15:30
Duration: 1.5 hours

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.**
6. **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.**

Materials provided

Answer booklets

Please note that the exam questions are printed on both sides of the exam paper.

This exam paper must not be removed from the exam room.

THIS PAGE HAS BEEN LEFT INTENTIONALLY BLANK

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 cubric time

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

      i) $\mathrm{O(n^3)}$

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

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

   (3 marks)

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

      i) $T(n) = 2000$

      ii) $T(n) = 24\ \ log_2\, n + 5n$

      iii) $T(n) = 4n^2 + \frac{2^n}{500} + 8 + \frac{1}{10}n^3$

   (3 marks)

2. a) In data structures and algorithms, what does the abbreviation **ADT** stand for?

   (1 mark)

   b) With the aid of a suitable diagram, briefly describe what is meant by the ADT **list**.

   (6 marks)

3. a) Write down the complete code, including appropriate comments, for a generic Java interface to model the ADT **queue**.

(10 marks)

b) The Java Collections Framework (JCF) includes the interface `java.util.Queue` which models the ADT **queue**. Name TWO concrete implementation of the ADT **queue** in the JCF.

(2 marks)

4. Consider the following Java code for method `displayArray`, which displays the specified contents of an integer array:

Beginning of code

```
1  public void displayArray(int[] array, int first, int last)
2  {
3      System.out.print(array[first] + " ");
4      if (first < last) {
5         displayArray(array, first + 1, last);
6      }
7      System.out.println();
8  }
```

End of code

Making use of a suitable **Big-O** notation, state the time complexity of method `displayArray`. Justify your answer.

(6 marks)

5. a) Suppose you need to use **binary search** to determine whether a given element is in a collection containing 8 elements. Using a suitable diagram and making use of a suitable example, illustrate the **binary search** algorithm.

(5 marks)

b) Can the **binary search** algorithm be used to determine whether the name `Faith` is in the following list of names? Justify your answer.

Dottie Aimee Neve Evie Alba

(4 marks)

6. a) Making reference to the term **hash function**, briefly explain how data are stored in a **hash table**. (3 marks)

   b) Briefly describe what is meant by **collision** in a **hash table**. (3 marks)

   c) **Chaining** is a way to resolve collision in a hash table. Making use of a suitable diagram, outline ONE implementation of chaining. (4 marks)

END OF SECTION A

Section B — Answer TWO questions in this section

Each question in this section carries 25 marks.

7. a) Briefly describe what is meant by the ADT **set**. (5 marks)

   b) **Array**, **linear linked structure** and **hash table** are three types of data structures that can be used to implement an ADT. Which ONE of these data structures is best-suited for implementing a **bounded set**? Making reference to the characteristics of these data structures and the time efficiency of typical **bounded set** operations, justify your answer. (10 marks)

   c) Consider a standalone, single-user Java application which acts as a telephone directory for businesses. This Java application loads all data into the computer's main memory when it starts up. A user of this application:

   - TYPICALLY searches for the contact information (such as telephone number and address) of a company by the company name, and
   - OCCASIONALLY prints the contact information of all companies, with the contact information listed by company names alphabetically.

   Assume that the company names in this Java application are unique:

   i) Which ADT is suitable for modelling the data for the contact information in this application so as to support an efficient search of contact information? (1 mark)

   ii) Which class of collection object defined in the Java Collections Framework (JCF) is suitable for modelling the ADT in part (i)? (2 marks)

   iii) Making reference to your answer in parts (i) and (ii), explain how you would model the data in this business telephone directory so as to support BOTH of the above operations efficiently.

   Your explanation should include a comment on the resulting time efficiency of each operation. You may use a UML class diagram to aid your explanation, but it is not essential.

   (7 marks)

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

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

(2 marks)

(question continues on next page. . .)

(Question 8 continued. . .)

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

(5 marks)

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

(6 marks)

d) Briefly describe what is meant by the programming technique **recursion**.

(2 marks)

e) Method `size` in class `LinkedStack` returns the number of elements in the `stack`. Write the required modifications of class `LinkedStack` to implement this method using **recursion**.

You must clearly specify the location(s) of your modifications using the given line numbers.
An implementation which does not use recursion will not receive any mark.

(10 marks)

9. a) Briefly describe what is meant by a **binary search tree**. (5 marks)

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

```
1  public class BinarySearchTree<E> {
2
3      // the root node of this binary search tree
4      private BinaryTreeNode<E> root;
5
6      // Constructor: Creates an empty tree.
7      public BinarySearchTree() {
8          root = null;
9      }
10
11     /* Inner class */
12     protected static class BinaryTreeNode<V> {
13         // Implementation details omitted.
14     }
15 }
```

(question continues on next page. . .)

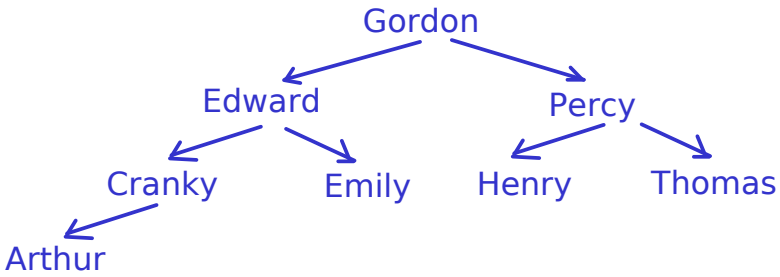
(Question 9 continued. . .)

There is a mistake in the header line of the given class `BinarySearchTree` which makes it fail to model an important characteristic of a **binary search tree**. Furthermore, all `BinarySearchTree` objects are expected to support the operation of an enhanced `for` statement.

Correct the header line of the given class `BinarySearchTree` so as to address the two stated issues.

(4 marks)

c) Consider the following **tree**:



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

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

iii) How many **parent** nodes are there on the tree? (1 mark)

iv) State the value at the **root** node of the tree. (1 mark)

v) Can the above tree be considered as a **balanced tree**? Making reference to the definition of a **balanced tree**, justify your answer.

(6 marks)

vi) Explain how the method `remove` would work when removing the **root** node from the tree. Show the resulting tree after the node removal.

(6 marks)

END OF EXAMINATION PAPER