Aston University

School of Engineering and Applied Science

CS2310 - Data Structures and Algorithms with Java

Second Year Examination

CLOSED BOOK

Date: 11th January, 2016
Time: 09:30 – 11:30
Duration: 2 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**

Materials provided

1. **Answer booklets**

This exam paper cannot be removed from the exam room

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 $\mathrm{O(n^2)}$ time

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

      iii) executes in $\mathrm{O(2^n)}$ time

   (3 marks)

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

      i) $T(n) = 450$

      ii) $T(n) = 3n^3 + 55n$

      iii) $T(n) = 20^{100} + \frac{1}{100}n + 2^n$

   (3 marks)

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

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

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

Your explanation should include a justification for the chosen **Big-O** *notation and make reference to the number of calls to* `fibonacci` *at Line 5.*

(6 marks)

3. a) Making reference to the four typical operations of a **map**, describe what is meant by the Abstract Data Type (ADT) **map**.

(6 marks)

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

(1 mark)

4. Consider two search algorithms: **sequential search** and **binary search**.

a) Which ONE of these two search algorithms is more efficient in determining whether the name `Lucy` is in the following list of names?

Caspian Digory Edmond Lucy Peter Susan

(2 marks)

b) Making reference to how **sequential search** and **binary search** work and the assumption each of them make on the nature of the **search pool**, explain your answer to part (a).

(10 marks)

5. Consider the following array-based partial implementation of a **bounded stack**:

```
1  public class ArrayStack<T>
2  {
3      private T[] contents;
4      private int top; // index to the next unoccupied array cell
5
6      // constructs a new empty stack
7      public ArrayStack(int capacity) {
8          top = 0;
9          contents = (T[]) new Object[capacity];
10     }
11
12     // returns the number of elements in the stack
13     public int size() {
14         return top;
15     }
16
17     // returns true if the stack is empty, otherwise false
18     public boolean isEmpty() {
19         return (size() == 0);
20     }
21
22     // adds the specified element to the stack
23     public void push(T element) {
24         // method details omitted
25     }
26 }
```

a) Write Java code for method `clear` in class `ArrayStack` which runs in **constant time**. Method `clear` sets the stack to empty. (3 marks)

b) Write Java code for method `pop` in class `ArrayStack` which runs in **constant time**. `pop` is a typical operation in a **stack** ADT. (5 marks)

c) Describe in some detail ONE common use of **stack**. (4 marks)

6. a) Making use of a diagram, briefly describe what is meant by a **queue** data structure and how it works.

(4 marks)

b) Briefly describe what is meant by a **priority queue** and how it works.

(3 marks)

END OF SECTION A

Section B — Answer TWO questions in this section

Each question in this section carries 25 marks.

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

(5 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.

(14 marks)

c) Indicate clearly what changes and additions need to be made to the following class `Employee` 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 field `gradePoint` and employees who have equal `gradePoint` are to be stored in alphabetical order of the field `name`.

```
1  public class Employee
2  {
3      private int gradePoint;
4      private String name;
5
6      public Employee(int gradePoint, String name) {
7          this.gradePoint = gradePoint;
8          this.name = name;
9      }
10
11     public int getGradePoint() {
12         return gradePoint;
13     }
14
15     public String getName() {
16         return name;
17     }
18 }
```

(6 marks)

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

```
1  public class LinkedSet<E> {
2    // the first node in the linked structure
3    private LinearNode<E> first;
4    private int size;
5
6    public LinkedSet() {
7      first = null;
8      size = 0;
9    }
10
11   public boolean add(E element) {
12     // definition omitted
13   }
14
15   public E removeRandom() {
16     // definition omitted
17   }
18
19   public boolean contains(E element) {
20     // definition omitted
21   }
22
23   public boolean isEmpty() {
24      return (first == null);
25   }
26
27   // Other method definitions omitted
28
29   private static class LinearNode<T> {
30     public T element;
31     public LinearNode<T> next;
32
33     public LinearNode(T element) {
34       this.element = element;
35       this.next = null;
36     }
37   } // END OF CLASS LinearNode
38
39 } // END OF CLASS LinkedSet
```

(question continues on next page. . .)

(Question 8 continued. . .)

a) Making reference to THREE typical operations of **set**, explain what is meant by the Abstract Data Type (ADT) **set**. (5 marks)

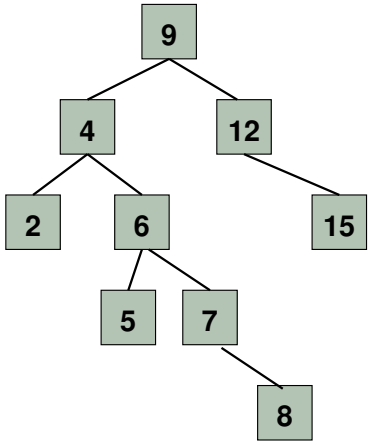
b) Write Java code for method `add` in class `LinkedSet`. (8 marks)

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 set**? (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) Name ONE concrete implementation of **set** in the Java Collections Framework (JCF). (1 mark)

9. 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) How many nodes on the tree are **leaf** nodes? (1 mark)

iv) State the kind of tree that best describes the characteristics of the given tree. (2 marks)

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

i) **pre-order traversal** (2 marks)

ii) **in-order traversal** (2 marks)

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

```
1  public class BinaryTree<T> {
2      // the root node of this tree
3      private BinaryTreeNode<T> root;
4
5      /** Constructor: Creates an empty tree. */
6      public BinaryTree() {
7          root = null;
8      }
9
10     // Other methods omitted
11 }
```

i) Write Java code to define a **generic static nested class** `BinaryTreeNode` in class `BinaryTree` to model a node in a **binary tree**.

Class `BinaryTreeNode` *should include* THREE **public** *fields and* ONE *constructor for creating a* **leaf** *node. No* **getter** *and* **setter** *methods are required for class* `BinaryTreeNode`*.*

(8 marks)

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

(8 marks)

END OF EXAMINATION PAPER