Aston University

School of Engineering and Applied Science

CS2310 Data Structures and Algorithms with Java

Second Year Examination

CLOSED BOOK

Date:
Time:
Duration: **1.5 hours**

Instructions to Candidates

1. **Section A is worth 50 marks in total.**
2. **Answer ALL questions in section A.**
3. **Section B contains THREE questions, EACH worth 25 marks.**
4. **Answer TWO questions ONLY in section B.**
5. **Use of calculators is not allowed.**

Materials provided

1. **Answer booklets**

This exam paper cannot be removed from the exam room

Section A — Answer ALL questions in this section

1. a) Using the **Big-O notation**, write down suitable expressions for describing the time complexity of algorithms that execute in:

i) constant time,
ii) logarithmic time,
iii) cubic time,
iv) log-linear time.

(4 marks)

b) Making use of a suitable **Big-O notation**, briefly explain the time efficiency of the following Java method which computes the sum from 1 to n.

```
public int sum(int n){

        return n*(1+n)/2;
}
```

(2 marks)

c) State the time complexity using **Big-O** notation for EACH of the following **growth functions**:

i) $T(n) = n + 1024$

ii) $T(n) = n^2 + 800000n + 1$

iii) $T(n) = 10n + n \log_2 n$

iv) $T(n) = 2^n + n^3$

(4 Marks)

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

(2 Marks)

b) **Stack** is one of the popular ADTs. List THREE other popular ADTs.

(3 Marks)

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

(3 marks)

b) **Chaining** is a way to resolve collisions in a hash table. Making use of a suitable diagram, outline ONE implementation of chaining.

(4 marks)

c) The methods **hashCode** and **equals** are both defined in the class java.lang.Object. Assume a class overriding **equals** but not **hashCode**, state briefly what problems this could cause.

(3 marks)

4. a) Making use of Java **generics**, write a **generic** Java interface called **StackADT**. This interface should specify the basic operations of a stack.

(6 marks)

b) **StackADT** can be implemented using **linked list** or **Array**. If a **bounded stack** is to be implemented, which of the two structures would you and why?

(3 marks)

c) Identify TWO practical uses of the **stack** data structure.

(4 marks)

5. a) Briefly explain what is meant by a **perfect n-ary tree**.

(2 marks)

b) Making use of a suitable diagram, briefly explain what is meant by a **balanced binary tree**.

(4 marks)

c) Consider the following incomplete definition of class **TreeNode**. This class models a node in an **n-ary tree** data structure, where n is unknown.

```
public class TreeNode<E> {

// Fields


/** Constructor to create a leaf TreeNode */


// method definitions

}
```

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

(2 marks)

ii) Write Java code to define a constructor for class `TreeNode`. This constructor enables the creation of a new `TreeNode` object to model a leaf node in an **n-ary tree** data structure.

(4 marks)

Section B — Answer TWO questions in this section

Each question in this section carries 25 marks.

6. a) Consider the following list of names:

Joe Michel Amy Ian Jonah Esra

i) Briefly explain the **linear search** and **binary search** methods.

(4 marks)

ii) Which search method should be used for determining whether a particular name is present in the above list?

(2 marks)

iii) Making use of the above list of names, illustrate the execution steps of the **selection sort algorithm** using a diagram.

(7 marks)

b) Suppose you need to use **binary search** to determine if a given element is in a collection containing 32 elements. State how many comparisons between elements the binary search process will need to carry out in order to determine that the given element does not exist in the collection. Justify your answer

(4 marks)

c) **Quick sort** orders a list of values by splitting the list into two partitions around one element, then sorting each partition. This work is divided into two methods:

- `quickSort`: performs the recursive algorithm
- `findPartition`: rearranges the elements into two partitions.

i) Consider the following incomplete code extract. Write out the missing lines of code in the method `quickSort.`

(6 marks)

(Question continues on next page……)

(Question 6 continued……)

```
public static <T extends Comparable<T>>
   void quickSort (T[] data, int min, int max){

      int indexofpartition;

      if (max > min) {
      // Please write the code needed here.

     }
}



private static <T extends Comparable<T>>
  int findPartition (T[] data, int min, int max) {

/**implementation details omitted */

}
```

ii) Give a suitable **Big-O** notation for describing the time complexity of the **quick sort** algorithm.

(2 marks)

7. a) Write a Java code to define a **generic** Java interface `IndexedListADT`**.** This interface should specify at least FIVE typical operations for an **indexed list**. Instances of classes which implement this `IndexedListADT` must also support the operation of an **enhanced for** statement.

(8 marks)

b) Briefly describe TWO distinct ways to implement an **indexed lis**t.

(4 marks)

c) Explain what is meant by an **ordered list**, highlighting the differences from an **indexed list**, in particular in terms of their typical operations.

(6 marks)

d) Consider the following definition for class `ActorRole`. Class `ActorRole` models a character in a film which an actor plays.

```java
public class ActorRole
{
  private String name;  // the name of an actor
  private String role;  // the role played the actor
  private int billingOrder;

  public String getName() {
   return name;
  }

  public String getRole() {
    return role;
  }

  public int billingOrder() {
    return billingOrder;
  }

 // Details of constructor, other method and fields
 // omitted
}
```

(Question continues on next page……)

(Question 7 continued……)

Suppose a film database application is to use an **ordered list** to store a collection of `ActorRole` objects for each film. A brochure for each film is to be printed with the actors listed in their billing order and then alphabetically by their role in the film. Write out the necessary modifications for class `ActorRole` to enable such a film brochure to be prepared.

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

(7 marks)

8. Consider the following definition for class `ArrayQueue`. Class `ArrayQueue` models the basic operations of a queue using an array.

```
import exceptions.EmptyQueueException;
import exceptions.FullQueueException;

public class ArrayQueue<E> {
  private E[] queue;
  // the location of the last element in the queue
  private int last;

  /** Create an empty queue with a specified capacity */
  public ArrayQueue(int capacity)  {
       queue = (E[]) (new  Object[capacity]);
       last = -1;
  }

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

  public E dequeue() throws EmptyQueueException {
       // definition omitted
  }

  public  boolean isEmpty() {
       // definition omitted
  }

  public  boolean isFull() {
       // definition omitted
  }

  public int size() {
       // definition omitted
  }
}
```

Class `ArrayQueue` and the package exceptions exist in the SAME folder within a file system. The exception class `EmptyQueueException` has the constructor `EmptyQueueException()`and the exception class `FullQueueException` contains the constructor `FullQueueException()`.

(Question continues on next page……)

(Question 8 continued……)

a) **Enqueue** and **dequeue** are two typical operations for a queue. Making use of a diagram, briefly explain what EACH of these operations does.

(5 marks)

b) Suppose the class `ArrayQueue` has been fully implemented and the following code is executed:

```
1   ArrayQueue<Character> q = new ArrayQueue<Character>(3);
2   q.enqueue('f');
3   q.enqueue('e');
4   q.enqueue('d');
5   q.enqueue(q.dequeue());
```

i) After line 1 has finished its execution, what is the state of the `ArrayQueue` object q?

(2 marks)

ii) After the method call at line 5 has finished its execution, what is the state of the `ArrayQueue` object q?

(3 marks)

c) Briefly describe the role and purpose of the identifier **`E`** at line 4 in the given definition of `ArrayQueue`.

(4 marks)

d) Write an implementation of the method `size` for the given definition of `ArrayQueue`. This method must execute in constant time.

(3 marks)

e) Write an implementation of the method **dequeue** for the given definition of `ArrayQueue`.

(8 marks)

END OF EXAMINATION PAPER