RESIT EXAMINATIONS 2012/13

Advanced Object-Oriented Programming

TIME ALLOWED: Two hours

INSTRUCTIONS TO CANDIDATES

Answer four questions.

If you attempt to answer more questions than the required number of questions (in any section), the marks awarded for the excess questions answered will be discarded (starting with your lowest mark).
1. Below is a Maude specification of ‘Sorted Lists’: lists of integers in increasing order. Operations add and empty give a list structure; operation insert will insert an integer at the appropriate point in a sorted list.

\textbf{fmod} SORTED-LIST \textbf{is}

\begin{itemize}
  \item protecting INT .
  \item sort SortedList .
  \item op empty : \rightarrow SortedList [ctor ] .
  \item op add : Int SortedList \rightarrow SortedList [ctor ] .
  \item op insert : Int SortedList \rightarrow SortedList .
  \item vars I J : Int .
  \item var L : SortedList .
  \item eq \hspace{0.5cm} \text{insert}(I, \text{empty}) = \text{add}(I, \text{empty}) .
  \item cq \hspace{0.5cm} \text{insert}(I, \text{add}(J, L)) = \text{add}(I, \text{add}(J,L)) \text{ if } I \leq J .
  \item cq \hspace{0.5cm} \text{insert}(I, \text{add}(J, L)) = \text{add}(J, \text{insert}(I, L)) \text{ if } I > J .
\end{itemize}
\textbf{endfm}

(a) Give a Java implementation of SortedLists, using linked lists. Marks will be awarded for correctly implementing:
\begin{itemize}
  \item i. the linked list structure; \hspace{1cm} [5 marks]
  \item ii. a constructor corresponding to the operation ‘empty’; \hspace{1cm} [2 marks]
  \item iii. method ‘add’; and \hspace{1cm} [3 marks]
  \item iv. method ‘insert’. \hspace{1cm} [6 marks]
\end{itemize}

(b) Why would it be appropriate not to implement ‘add’ as a public method? \hspace{1cm} [5 marks]

(c) Linked lists are one way of implementing lists; it is also possible to implement lists using arrays. Give one advantage and one disadvantage of each of these two ways of implementing lists. \hspace{1cm} [4 marks]
2. For this question, we consider an application where we need to store integer data (for example, exam results) with string ‘keys’ (for example, student names): each string will have an associated integer (for example, the exam mark for a given student). We call a pair consisting of a string and an integer a ‘KeyValuePair’, and we call a list of such pairs a ‘KeyValuePairList’.

A KeyValuePair is constructed by an operation, ‘makePair’, that takes a string and an integer as arguments; there are two ‘accessor’ operations: ‘getKey’, which returns the string key of a KeyValuePair, and ‘getValue’, which returns the integer value of a KeyValuePair.

KeyValuePairLists are constructed by means of a constant operation, ‘empty’, representing an empty list of KeyValues, and an operation, ‘add’, which takes a KeyValuePair and a KeyValuePairList and returns the KeyValuePairList formed by adding the given pair to the given list. In addition, there is an operation ‘get’ that takes a string key and a KeyValuePairList and returns the integer value associated with the given key. Specifically, the operation searches through the list to find a KeyValuePair with the given key, and returns the integer value for that KeyValuePair; if the list does not contain a KeyValuePair with the given key, the value -1 is returned.

(a) What, precisely, is meant by the term Abstract Data Type? [6 marks]

(b) Give a Maude specification of the abstract data type KeyValuePair. (One mark will be given for correctly specifying the sort KeyValuePair; three marks will be given for correctly specifying the type of each of the three required operations, and four marks for correctly specifying their required behaviour by giving two equations.) [8 marks]

(c) Give a Maude specification of the abstract data type of KeyValuePairLists. (Two marks for correct Maude syntax; three marks for specifying the required operations, and six marks for the equations.) [11 marks]
3. The following class implements bounded queues of integers: a queue is a first-in, first-out list that can store at most five integers. The class has a method remove(int) that removes the given number of elements from the queue.

```java
public class Queue {
    private int[] items = new int[5];
    private int first = -1; // index of first item in queue
    private int last = 0;   // 1 + index of last item

    public boolean isEmpty() {
        return last <= 0;
    }

    public int get() {
        int val = items[first];
        for (int i = 0; i+1 < last; i++) {
            items[i] = items[i+1];
        }
        last--;
        if (last == 0) first--;
        return val;
    }

    public void add(int i) {
        if (last == 0) first++;
        items[last++] = i;
    }

    public void remove(int i) {
        if (i <= 0) return;
        get();
        remove(i - 1);
    }

    public static void main(String[] args) {
        Queue s = new Queue();
        s.add(3);
        s.remove(2);
    }
}
```

(a) Briefly describe the function of the method-call stack in the Java interpreter. [3 marks]

(b) What happens when the main method in the Queue class is executed? Describe the state of the method-call stack during execution of the main method. [5 marks]
(c) Briefly describe the differences between ‘checked’ and ‘unchecked’ exceptions. [4 marks]

(d) Write a checked exception class EmptyQueueException, and modify the get() method so that it throws an EmptyQueueException if the queue is empty. What other changes would be necessary to the class? [7 marks]

(e) With these changes, is the property $0 \leq \text{last}$ a class invariant? Justify your answer. [6 marks]
4. Consider the following program, which creates two threads sharing a Queue instance (using the class from Question 3). The goal of the program is simply to print out roughly 200 numbers, without throwing any exceptions.

```java
class ThreadTest implements Runnable {
    private Queue queue;
    ThreadTest(Queue s) {
        queue = s;
    }

    public void run() {
        for (int i = 0; i < 200; i++) {
            if (queue.isEmpty()) {
                queue.add(i);
            } else {
                System.out.println(queue.get());
            }
        }
    }

    public static void main(String[] args) {
        Queue s = new Queue();
        Thread t1 = new Thread(new ThreadTest(s));
        Thread t2 = new Thread(new ThreadTest(s));
        t1.start();
        t2.start();
    }
}
```

(a) In the context of multi-threaded programs, what is meant by *interference*? [3 marks]

(b) Describe how interference might arise when the ThreadTest main method is executed. [8 marks]

(c) Describe how *synchronization* can be used to prevent interference. [8 marks]

(d) How would you modify the ThreadTest or Queue class to prevent interference? [6 marks]
5. The Queue class in Question 3 stores integers. Java uses generic classes to allow the implementation of data structures that can store instances of arbitrary classes.

(a) Modify the Queue class to give a generic implementation of queues: i.e., use a generic type variable E so that the values stored in the queue all belong to the class E.

[6 marks]

(b) Using your answer to part (a), give Java code that creates an instance of a queue class that can store instances of class Integer, and add integers 3 and 7 to the queue.

[3 marks]

(c) Write a generic method that takes a queue of queues (i.e., the parameter is of type Queue<Queue<E>>) so that every element in the queue is itself a queue), and returns a queue containing all the elements in each queue in the given queue (so that all elements in the returned queue belong to the generic class E — it does not matter what order the elements occur in).

[6 marks]

(d) Java implements generic types by a process called erasure, which removes generic type parameters. What would the results of erasure be for:

i. the Queue class in your answer to part (a);

[4 marks]

ii. your code in your answer to part (b);

[2 marks]

iii. the generic method in your answer to part (c)?

[4 marks]