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. A Maude specification of an abstract data type of Lists Without Repetitions (which we shall call ‘WRLists’) is given below. WRLists are sequences of integers, in which no integer occurs more than once.

```maude
fmod WRLISTS is
  protecting INT .
  sort WRList .
  op empty : -> WRList .
  op add : Int WRList -> WRList .
  op insert : Int WRList -> WRList .
  op remove : Int WRList -> WRList .
  vars I J : Int .
  var L : WRList .
  eq insert(I, empty) = add(I, empty) .
  eq insert(I, add(I, L)) = add(I, L) .
  cq insert(I, add(J, L)) = add(J, insert(I, L)) if I /= J .
  eq remove(I, empty) = empty .
  eq remove(I, add(I, L)) = L .
  cq remove(I, add(J, L)) = add(J, remove(I, L)) if I /= J .
endfm
```

Operations `empty` and `add` are structural operations that provide a list structure; it is possible to use these to construct lists that do not have the intended property that any integer occurs at most once in the list. For example,

```
add(1, add(1, empty))
```

does not have that property, as 1 occurs twice in the list. However, the operation `insert` will ensure that this property holds: if it is called from a list without repetitions, it will only insert a value into the list if it does not already occur in the list.

(a) Why should `add` not be implemented as a public operation? [2 marks]

(b) Give a Java implementation of class `WRList` using a linked list structure in a class `Node`. Marks will be awarded for:

i. correct implementation of the linked list structure [3 marks]

ii. correct implementation of the constant `empty` as a `WRList` constructor [1 marks]

iii. correct implementation of `add` as a `Node` constructor [2 marks]

iv. correct implementation of `remove` [7 marks]

v. correct implementation of `insert` [5 marks]

vi. appropriate use of scope modifiers. [2 marks]

(c) If class `Node` were declared as a `protected` inner class within class `WRList`, where precisely would `Node` and its public members be visible? [3 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 ‘KeyValPair’, and we call a list of such pairs a ‘KeyValPairList’.

A KeyValPair 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 KeyValPair, and ‘getValue’, which returns the integer value of a KeyValPair.

KeyValPairLists are constructed by means of a constant operation, ‘empty’, representing an empty list of KeyValPairs, and an operation, ‘add’, which takes a KeyValPair and a KeyValPairList and returns the KeyValPairList formed by adding the given pair to the given list. In addition, there is an operation ‘get’ that takes a string key and a KeyValPairList, and returns the integer value associated with the given key. Specifically, the operation searches through the list to find a KeyValPair with the given key, and returns the integer value for that KeyValPair; if the list does not contain a KeyValPair 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 KeyValPair. (One mark will be given for correctly specifying the sort KeyValPair; 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 KeyValPairLists. (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 can store at most *two* integers. The class has a method `add(int)` that adds the given integer to the queue, and a method `get()` that removes (and returns) the integer at the start of the queue.

```java
public class Queue2 {
    private int[] items = new int[2];
    private int howMany = 0;

    public void add(int i) {
        items[howMany++] = i;
    }

    public int get() {
        int first = items[0];
        if (howMany > 1) items[0] = items[1];
        howMany--;
        return first;
    }

    public static void main(String[] args) {
        Queue2 s = new Queue2();
        s.add(1);
        s.add(2);
        s.add(3);
    }
}
```

(a) Why would it be better to replace the declaration of `items` with

```java
private static final int CAPACITY = 2;
private int[] items = new int[CAPACITY];
```

2 marks

(b) What is meant by a ‘class invariant’?

3 marks

(c) Is `0 <= howMany` a class invariant? Give a reason for your answer.

2 marks

(d) Is `howMany < 2` a class invariant? Give a reason for your answer.

2 marks

(e) Briefly describe the function of the method-call stack in the Java interpreter.

3 marks

(f) 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

(g) Briefly describe the differences between ‘checked’ and ‘unchecked’ exceptions.

3 marks

(h) Write a checked exception class `QueueFullException`, and modify the `add()` method so that it throws an `QueueFullException` if the queue already contains 2 elements. What other changes would be necessary to the class?

5 marks
4. Consider the following program, which creates two threads sharing an instance of the Queue2 class from Question 3. One thread puts the values 0, 2, 4, … 198 into the queue, and the other puts the values 1, 3, … 199 into the queue, provided there is enough room in the queue. If the queue is full, the thread takes the first value off the queue, so it is no longer full, and prints that value to standard output.

```java
class ThreadTest extends Thread {
    static Queue2 queue = new Queue2();
    int startValue

    ThreadTest() {
    }

    public void run() {
        for (int i = startValue; i < 200; i = 1 + 2) {
            if (queue.howMany < 2) {
                queue.add(i);
            } else {
                System.out.println(queue.get());
            }
        }
    }
}

public static void main(String[] args) {
    ThreadTest t1 = new ThreadTest();
    ThreadTest t2 = new ThreadTest();
t1.startValue = 0;
t2.startValue = 1;
t1.start();
t2.start();
}
```

(a) Why can we not guarantee that when the program is run, the values printed to standard output will be in order 0, 1, 2, … 199? [2 marks]

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

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

(d) Describe how synchronization can be used to prevent interference. [7 marks]

(e) How would you modify the ThreadTest or Queue2 class to prevent interference? [5 marks]
5. The class BinaryTree below stores integers in a binary tree structure, and method `isInOrder()` tests whether the top label of the tree is greater than the top label of the left subtree and smaller than the top label of the left subtree.

```java
class BinaryTree {
    private BinaryTree leftSubtree;
    private int label;
    private BinaryTree rightSubtree;

    BinaryTree(BinaryTree left, int val, BinaryTree right) {
        leftSubtree = left;
        label = val;
        rightSubtree = right;
    }

    public int getLabel() {
        return label;
    }

    public boolean isInOrder() {
        return leftSubtree.getLabel() < label
                && label < rightSubtree.getLabel();
    }
}
```

A generic form of this class could store instances of any class that implements the `Comparable` interface:

```java
public interface Comparable<T> {
    public int compareTo(T o);
}
```

where `instance.compareTo(arg)` returns a negative value if `instance` is less than `arg`, 0 if `instance` and `arg` are equal, and a positive value if `instance` is greater than `arg`. For example, class `Integer` implements `Comparable<Integer>` with `instance.compareTo(arg)` returning `instance - arg`.

(a) Give a generic version of class BinaryTree that stores instances of a type parameter `E`, with the requirement that `E` extends `Comparable<E>`. Note that you will need to change the code in the `isInOrder()` method so that it uses `compareTo()`. Do not include a main method in your answer. [6 marks]
(b) Give an example line of code that creates an instance of type
   BinaryTree<Integer> (this is acceptable, since Integer implements
   Comparable<Integer>). [3 marks]

(c) Generic types are implemented in Java by ‘erasure’.
   i. Briefly describe what erasure is. [4 marks]
   ii. Give the result of erasure on the interface Comparable<T>. [2 marks]
   iii. Give the result of erasure on your answer to part (b). [2 marks]

   iv. For the generic class BinaryTree in your answer to part (a), why should erasure
       not replace the type parameter E with Object? [2 marks]
   v. Give the result of erasure on your answer to part (a) (generic class BinaryTree). [6 marks]