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 ‘Key-Value’ Pairs and Lists of Pairs. Strings are used as keys and integers as values.

```
mod PAIRS is
  protecting STRING .
  protecting INT .
  sort Pair .
  op makePair : String Int -> Pair .
  op(getString : Pair -> String .
  op getInt : Pair -> Int .
  sort PairList .
  op empty : -> PairList .
  op add : Pair PairList -> PairList .
  op get : String PairList -> Int .

var S : String .
var I : Int .
var P : Pair .
var PL : PairList .
eq getString(makePair(S, I)) = S .
eq getInt(makePair(S, I)) = I .
eq get(S, empty) = -1 .
eq get(S, add(P, PL)) = getInt(P) if getString(P) == S .
eq get(S, add(P, PL)) = get(S, PL) if getString(P) /= S .
endm
```

The operation ‘makepair’ constructs a Pair from a given String and integer, and these can be accessed using the operations ‘getString’ and ‘getInt’, respectively. Lists of pairs are built up using the constant ‘empty’ and the operation ‘add’. Operation ‘get’ takes a string and a list as arguments: if it finds in the list a pair that has the given string as key, then it returns the integer from that pair; otherwise, if there is no such pair in the list, it returns a default value of -1.

(a) Give a Java implementation of the sort Pair. [6 marks]

(b) Give an implementation of the sort PairList, using linked lists. Marks will be awarded for correctly implementing:
   i. the linked list structure; [4 marks]
   ii. a constructor corresponding to the operation ‘empty’; [2 marks]
   iii. method ‘add’; and [3 marks]
   iv. method ‘get’. [5 marks]

(c) Say why your implementation of the operation ‘get’ behaves in the way specified by the two conditional equations in the Maude specification. [5 marks]
2. The Abstract Data Type of Count Lists is a list of pairs; each pair in the list has an integer (which we call the ‘value’) and another integer (which we call the ‘count’); the idea is that the ‘count’ integer records how often the value occurs in the list. An example application might be to read in a sequence of integer values, perhaps recording the grades of individuals in a class, e.g.,

$$2, 7, 7, 5, 9, 3, 6, 7, 2, 5$$

(1)

and, for each grade value, record how often that grade occurs in the given data. For the example data above, we should have the following Count List (where pairs are grouped in brackets):

$$(2, 2), (3, 1), (5, 2), (6, 1), (7, 3), (9, 1).$$

Note that grades that don’t occur in the data, such as 4, are not listed.

As an Abstract Data Type, Count Lists have one constant, representing the empty list (no data); and the following operations:

- **add** a Pair (consisting of two integer values, a value and its count) to the list;
- **insert** a given integer value into the list (if it already occurs in the list, increment the count value; otherwise, add a new pair with the given value and a count of 1); and
- **getCount** — this takes an integer value as argument and returns the count of how often that value occurs in the list (0 if it doesn’t occur at all in the list).

Count Lists make use of a subsidiary Abstract Data Type of Pairs: this Abstract Data Type has three operations:

- a constructor that takes two integers and returns the Pair built from those values;
- **getValue**, returning the value component of the Pair; and
- **getCount**, returning the count.

(a) Specify the Abstract Data Type of Pairs in Maude. [6 marks]
(b) Specify Count Lists in Maude. [11 marks]
(c) Write a term in Maude that corresponds to the Count List recording all the data in (1) above. [3 marks]
(d) Sketch why the equations in your specification of Count Lists would allow you to conclude that the value 5 occurs twice in that list. [5 marks]
3. Consider the following program, which is intended to print out the strings "a1", "a2", ...
"a100", as well as the strings "b1", "b2", ...
"b100", using two threads, one to
print all the ‘a’ strings, and one to print out all the ‘b’ strings. The printing of the ‘a’ and
‘b’ strings can therefore be interleaved.

```java
class ThreadTest implements Runnable {
    static String printString = null;
    private String threadName;

    ThreadTest(String s) { threadName = s; }

    public void run() {
        int i = 0;
        while (i < 100) {
            if (printString == null) {
                printString = threadName + (++i);
            } else {
                System.out.println(printString);
                printString = null;
            }
        }
    }

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

Each thread uses a local variable `i` as a count variable, and the static variable `printString`
to store the strings to be printed. If `printString` is null, a thread will assign to it the
next string to be printed, incrementing its count variable; if `printString` is not null, it
will print out the string stored, and set `printString` to null, so it can be assigned the
next string to be printed.

(a) What is meant by ‘interference’ in multi-threaded programs? [3 marks]
(b) Describe how interference might arise in this example. [6 marks]
(c) Describe how synchronization is used in Java to prevent interference. [8 marks]
(d) Briefly describe the difference between a synchronized method and a synchronized
block of code. [3 marks]
(e) Describe how you could use a synchronized block of code in class ThreadTest in order
to prevent interference. [5 marks]
4. The following class implements a bounded queue of integers: the queue can store at most ten integers.

```java
public class Queue {
    private int[] items = new int[10];
    private int numItems = 0;

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

    public int next() {
        int n = items[0];
        shift();
        numItems--;
        return n;
    }

    private void shift() {
        for (int i=0; i+1 < numItems; i++) {
            items[i] = items[i+1];
        }
    }

    public static void main(String[] args) {
        Queue q = new Queue();
        System.out.println("first in queue is: "+ q.next());
        q.add(24);
    }
}
```

(a) Describe the state of the stack of method calls as the main method is executed. What Exception would you expect to be thrown, and what would the stack trace be? [8 marks]

(b) i. Write a checked Exception class EmptyQueueException, and modify the next() method in class Queue so that it throws an EmptyQueueException if the method is called when the queue is empty. What other changes would be required for Queue.java to successfully compile? [7 marks]

   ii. Now what happens when the main method is run? [4 marks]

(c) With the modification of part (b)(i) above, is 0 <= numItems a class invariant for Queue? Justify your answer. [6 marks]
5. The Queue class in Question 4 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`. For part (c) below, you will also need to add a boolean method `isEmpty()`, which returns true if the queue is empty; false otherwise. [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]