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 ‘search’ trees of integers. A search tree is a binary tree
that is either empty (written as the constant ‘null’), or is of the form node(T1, I, T2),
where I is an integer (we call it the ‘node value’ of the tree) and T1 and T2 are search trees
(we call them the left and right subtrees of the tree), with the property that all the integers
in the left subtree T1 are less than I, and all the integers in the right subtree T2 are greater
than I — moreover, all the nodes in T1 and T2 have the same property: all integers in the
left subtree are less than the node value, and all integers in the right subtree are greater than
the node value.

\[
\text{mod SEARCH-TREES is}
\]

\[
\begin{align*}
\text{protecting} & \text{ INT} . \\
\text{sort} & \text{SearchTree} . \\
\text{op null} : & \rightarrow \text{SearchTree} . \\
\text{op node} : & \text{SearchTree Int SearchTree} \rightarrow \text{SearchTree} . \\
\text{op height} : & \text{SearchTree} \rightarrow \text{Int} . \\
\text{op insert} : & \text{SearchTree Int} \rightarrow \text{SearchTree} . \\
\text{vars} & I J : \text{Int} . \\
\text{vars} & T1 T2 : \text{SearchTree} . \\
\text{eq} & \text{height(null)} = 0 . \\
\text{eq} & \text{height(node(T1, I, T2))} = \max(\text{height(T1)}, \text{height(T2)}) + 1 . \\
\text{eq} & \text{insert(I, null)} = \text{node(null, I, null)} . \\
\text{eq} & \text{insert(I, node(T1, I, T2))} = \text{node(T1, I, T2)} . \\
\text{cq} & \text{insert(I, node(T1, J, T2))} = \text{node(insert(I, T1), J, T2)} \text{ if } I < J . \\
\text{cq} & \text{insert(I, node(T1, J, T2))} = \text{node(T1, J, insert(I, T2))} \text{ if } I > J . \\
\end{align*}
\]

endm

Operations null and node are structural operations that provide a binary tree structure; it
is possible to use these to construct trees that do not have the intended property that all
integers in left subtrees are less than the node value and all integers in right subtrees are
greater than the node value. For example,

\[
\text{node(node(null, 3, null), 2, node(null, 1, null))}
\]

does not have that property, as 3 is in the left subtree, but is not less than 2. However, the
operation insert will ensure that this property holds: if it is called from a search tree, then
the new value will be inserted at the appropriate point.

(a) Give a Java implementation of classSearchTree that uses a class, TreeNode, to
provide the binary tree structure (in a similar way to how a class Node can be used
to provide a linked list structure). Class TreeNode should also provide methods
height() and insert(). Marks will be awarded for:

i. correct implementation of the binary tree structure using class TreeNode

[3 marks]
ii. correct implementation of the constant `null` as a `SearchTree` constructor [1 marks]

iii. correct implementation of `node` as a `TreeNode` constructor [2 marks]

iv. correct implementation of `height()` in class `SearchTree` [2 marks]

v. correct implementation of `insert` in class `SearchTree` [3 marks]

vi. correct implementation of `height()` in class `TreeNode` [2 marks]

vii. correct implementation of `insert()` in class `TreeNode` [7 marks]

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

(b) If class `TreeNode` were declared as a `protected` inner class within class `SearchTree`, where precisely would `TreeNode` and its public members be visible? [3 marks]
2. Message Boards provide a forum for communication by maintaining lists of messages that have been sent by users, which can be viewed and added to by other users. Messages are usually organised by topic. Users can start a new topic, or view a list of all topics that have been started by other users. From this list they can choose a topic and view the messages in it. Users can also add a message to a particular topic. The list of topics is typically sorted by the time a topic was last added to, with topics that were most recently added to coming first. This question asks you to specify lists of Topics in Maude; for simplicity, we will not require that most-recently updated threads be moved to the start of the list.

(a) What exactly is an Abstract Data Type? [3 marks]

(b) Give a Maude specification of an abstract data type of Messages, where a Message consists of the name of the sender, and the text of the message, both of which are strings. There are three operations:
   - newMessage, which creates a message, given two strings (the sender’s name, and the message text);
   - getSender, which returns the name of a given message’s sender (as a string); and
   - getText, which returns the text of a given message (as a string).

newMessage is a structural operation; you will need equations to define the behaviour of getSender and getText: both take a Message as argument. [6 marks]

(c) Give a Maude specification of an abstract data type of Topics, where a Topic is a non-empty list of Messages (i.e., every Topic has at least one Message), and a Topic also has a title (which is a String). The abstract data type has the following operations:
   - newTopic, which creates a Topic, given the Topic’s title (a String), and a Message (the first Message in the Topic);
   - addMessage, which takes a Topic and a message, and returns a Topic (the Topic with one more Message added to it); and
   - getTitle, which returns the title of the Topic (as a String).

NB: newTopic and addMessage are structural operations; you will need equations to define the behaviour of getTitle. [7 marks]

(d) Give a Maude specification of Topic Lists, which are lists of topics. There are two structural operations:
   - empty, a constant representing the empty list of Topics, and
   - add, which takes a Topic and a Topic List, and returns a Topic List.

In addition, there are the following operations:
   - addNewTopic, which takes a Message and a String (the title of the new Topic) and a Topic List, and creates a new Topic with the given Message and title, and adds it to the start of the given Topic List; and
   - addMessageToTopic, which takes a Message, a String (the title of the Topic to add the Message to), and a Topic List, and returns the Topic List, where the Topic with the given title has had the given Message added to it. If there is no Topic with the given title in the Topic List, then the operation leaves the list unchanged.

NB: you will need equations defining the behaviour of these two operations. [9 marks]
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. An implementation of bounded stacks is given below; a stack can hold at most five integers. The class has a method `remove(int)` that pops the stack a given number of times.

```java
public class Stack {

    private int[] values = new int[5];
    private int top = 0;

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

    public int pop() {
        return values[--top];
    }

    public void push(int i) {
        values[top++] = i;
    }

    public void remove(int i) {
        while (i > 0) {
            pop();
            i--;
        }
    }

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

(a) Briefly describe the ‘method-call stack’ in the Java interpreter (note that this is not related to the class Stack above). [3 marks]

(b) What happens when the `main` method in the Stack class is executed? Describe the state of the method-call stack during the execution of this `main` method. [5 marks]

(c) Briefly describe the differences between ‘checked’ and ‘unchecked’ exceptions. [4 marks]

(d) Write a checked exception class, `EmptyStackException`, and modify the `pop()` method in class `Stack` so that it throws an `EmptyStackException` if the stack is empty. What other changes would be necessary to the class? [7 marks]

(e) With these changes, is the property $0 \leq \text{top}$ a class invariant? Justify your answer. [6 marks]
5. Queues are First-In, First-Out lists: elements are removed from the ‘front’ of the queue and are added to the ‘end’ of the queue. A queue can be implemented by using an array, \texttt{values}, to store the elements in the queue, with an integer ‘pointer’, \texttt{startIndex}, indicating the index of the first element in the queue, and another, \texttt{endIndex}, whose value is one greater than the index of the last element in the queue. That is, the elements in the queue are all the values \texttt{values[i]}, where \texttt{startIndex} \leq i < \texttt{endIndex}.

Getting the element at the front of the queue can be done by returning the element stored in \texttt{values[startIndex]} and incrementing \texttt{startIndex} (or throwing an exception if the queue is empty, i.e., \texttt{startIndex == endIndex}).

Adding a value to the end of the queue is done by storing the value in \texttt{values[endIndex]} and incrementing \texttt{endIndex}, if \texttt{endIndex} is less than \texttt{values.length}. If \texttt{endIndex} is equal to \texttt{values.length}, then, if \texttt{startIndex} is greater than 0, room can be made to store the new element by moving all the values in the queue to the start of the array (and updating \texttt{startIndex} and \texttt{endIndex}). Otherwise, room will have to be made by creating a larger array (say, 10 more than \texttt{values.length}), copying all the elements in the queue to the larger array and assigning that array to the variable \texttt{values}, before storing the new element at the end.

\textbf{(a)} Implement a generic class \texttt{Queue\langle E\rangle}, of homogeneous queues that store elements of the parameter type \texttt{E}. Marks will be awarded for:

\begin{itemize}
  \item[i.] correct use of the parameter type \texttt{E} \hspace{1cm} [2 marks]
  \item[ii.] implementation of the method \texttt{getFirst()} that returns the first element in the queue and removes it from the queue, as described above \hspace{1cm} [5 marks]
  \item[iii.] Implementation of a checked Exception class that is thrown by \texttt{getFirst()} if the queue is empty \hspace{1cm} [3 marks]
  \item[iv.] implementation of the method \texttt{addToEnd(E v)} that adds the given (parameter) value to the end of the queue, as described above. \hspace{1cm} [9 marks]
\end{itemize}

\textbf{(b)} Generic types are implemented in Java by ‘erasure’. Briefly describe what erasure is, and illustrate your answer by giving the results of erasure on your \texttt{addToEnd(E v)} method from your answer to part (a) above. \hspace{1cm} [6 marks]