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.

\[\text{fmod SEARCH-TREES is}\]
\[\text{ protecting INT.}\]
\[\text{ sortSearchTree.}\]
\[\text{ op null : } \rightarrow \text{SearchTree [ctor].}\]
\[\text{ op node : SearchTree Int SearchTree } \rightarrow \text{SearchTree [ctor].}\]
\[\text{ op height : SearchTree } \rightarrow \text{Int.}\]
\[\text{ op insert : SearchTree Int } \rightarrow \text{SearchTree.}\]
\[\text{ vars I J : Int.}\]
\[\text{ vars T1 T2 : SearchTree.}\]
\[\text{ eq height(null) = 0.}\]
\[\text{ eq height(node(T1, I, T2)) = max(height(T1), height(T2)) + 1.}\]
\[\text{ eq insert(I, null) = node(null, I, null).}\]
\[\text{ eq insert(I, node(T1, I, T2)) = node(T1, I, T2).}\]
\[\text{ cq insert(I, node(T1, J, T2)) = node(insert(I, T1), J, T2) if I < J.}\]
\[\text{ cq insert(I, node(T1, J, T2)) = node(T1, J, insert(I, T2)) if I > J.}\]
\[\text{endfm}\]

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 class SearchTree 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` [6 marks]
viii. appropriate use of scope modifiers. [2 marks]

(b) Suppose the class `TreeNode` is declared as a `protected` inner class within the class `SearchTree`.

i. Where precisely would `TreeNode` and its public members be visible? [3 marks]

ii. Where would any private members of `TreeNode` be visible? [1 mark]
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 (ctor); 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: both newTopic and addMessage are structural operations (ctors); 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 (ctors):

- `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]
3. Consider the following class of binary trees. The method `isInOrder()` is intended to test 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();
    }

    public static void main(String[] args)
    {
        BinaryTree bt = new BinaryTree(null, 5, null);
        bt = new BinaryTree(bt, 7, null);
        System.out.println(bt.isInOrder());
    }
}
```

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

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

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

(d) Write a checked exception class, `NoSubtreeException`, and modify the `isInOrder()` method so that it throws a `NoSubtreeException` if either the left or right subtree is null. What other changes would be necessary to the class?
(e) What is meant by a ‘class invariant’? [2 marks]

(f) Modify the constructor of class BinaryTree so that it throws a checked exception if either:
   - the parameter left is not null and its label is not smaller than parameter val, or
   - the parameter right is not null and its label is not greater than parameter val.

With this modification, is the property that isInOrder() always returns true a class invariant for BinaryTree? [5 marks]
4. Consider the following two classes.

```java
class FriendFinderThread extends Thread {
    // number of FriendBook friends
    int numFriends = 0;

    public void run() {
        // join, then leave FriendBook
        FriendBook.join(this);
        try { Thread.sleep(10000); }
        catch (InterruptedException ie) {} 
        FriendBook.leave(this);
    }
}

class FriendBook {
    // list of FriendBook members
    static Vector<FriendFinderThread> members =
        new Vector<FriendFinderThread>();

    static void join(FriendFinderThread f) {
        // add a new friend to all existing members
        int size = members.size();
        for (int i = 0; i < size; i++) {
            members.elementAt(i).numFriends++;
        }
        f.numFriends = size; // new member’s friends
        members.add(f); // add to list of members
    }

    static void leave(FriendFinderThread f) {
        members.remove(f); // remove from list
        int size = members.size();
        for (int i = 0; i < size; i++) {
            members.elementAt(i).numFriends--;
        }
    }

    public static void main() {
        for (int n = 0; n < 100; n++) {
            new FriendFinderThread().start();
        }
    }
}
```
Class FriendBook maintains a list of its ‘members’; a FriendFinderThread instance becomes a member by calling the join() method. While it is a member of FriendBook, the numFriends field should be equal to the total number of members. When a new FriendFinderThread instance becomes a member, the join() method increments the numFriends field of all existing members; similarly, the leave() method decrements these fields.

(a) The method start() in class Thread will place a new thread in a pool of threads that are ‘ready’ to run under time-slicing. The Java interpreter will select a thread from this pool to ‘run’ (i.e., execute the thread). ‘Ready’ and ‘running’ are two of the possible states that threads may be in, and threads move between these states under the control of the Java interpreter’s time-slicing mechanism. What other states can threads be in, and how do they change from one state to another? [6 marks]

(b) One problem that might arise in the FriendBook program is ‘interference’. Briefly say what interference is, and describe how it might arise in this program. [6 marks]

(c) Describe how synchronization is used in Java to prevent interference. [6 marks]

(d) Briefly describe the difference between a synchronized method and a synchronized block of code. [3 marks]

(e) How would you use synchronization to prevent interference in the FriendBook program? Justify your answer. [4 marks]
5. The class BinaryTree in Question 3 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. A generic form of that 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) Modify the main method of BinaryTree so that it creates instances of type `BinaryTree<Integer>` (this is acceptable, since `Integer` implements `Comparable<Integer>`). [4 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) (the main method). [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? [1 mark]
   v. Give the result of erasure on your answer to part (a) (generic class BinaryTree). [6 marks]