Notes

This is not intended to be an exhaustive review of all of the problems that might appear on your final exam. Instead, it is a selection of problems that we have either done in class or on previous exams that require additional attention. For example: I didn’t include any specific question on Dijkstra’s algorithm; use the slides that have been done so well by others and that are available to you on Elms for that topic.

Please use these in the spirit in which they are offered. Try them first, in earnest. Answers may be provided later.

1. Using only the definitions provided by the References and standard Java statements, write the recursive function `append` that takes two `SList` and returns a new `SList` that contains the elements from both lists, in their original order. For example: `append( [1,2,3], [2,3,5] )` should return `[1,2,3,2,3,5]`. If `list1` is empty, then `list2` is returned, unmodified. Your implementation must be recursive and should be contained in one method.

```java
public static <T> SList<T> append( SList< T> list1, SList<T> list2 ) {
```
2. Study the following class definition:

```java
class Item {
    private String itemName;
    private float unitCost;
    private boolean isAvailable;
    ...
}
```

Two Items are equal only when their itemName and unitCosts properties are equal.

(a) Override the `hashCode()` method for the Item class:
```java
public int hashCode() {
```

(b) The class WishList looks like:
```java
class WishList implements Iterable<Item> {
    private List<Item> items;
    ...
}
```

Two WishLists are equal when the contain the same Items in the same order. Override the `hashCode()` method for the WishList class:
```java
public int hashCode() {
```
3. Examine the undirected graph below:

![Graph Diagram]

and use it to answer the following questions:

(a) Give the list of vertices visited in an ordered Breath-First traversal of the graph, starting from vertex B:

(a) __________

(b) Give the list of vertices visited in an ordered Depth-First traversal of the graph, starting from vertex B:

(b) __________

(c) Express the graph used in this question formally, i.e., as a set of Vertices and a set of Edges, as in

\[ G = (V, E) \]

List vertices by name, e.g., \( \{A, B, C, \ldots\} \) and edges as \( \{e_1(A, B), \ldots\} \).
4. Given any 2 BinaryTree, tree_1 and tree_2, write the sameShape predicate that returns true if and only if tree_1 and tree_2 have the same shape, i.e., each may be superimposed on the other with no apparent differences other than that the labeling of their nodes.

   public static <T> boolean sameShape( BTree< T > tree_1, BTree< T > tree_2 ) {
5. Study the classroom example below:

```java
class HelloWorldHello {
    String str1="Hello", str2="World";

    Thread T_1 = new Thread( new Thread() {
        public void run() {
            synchronized( str1 )
            synchronized( str2 )
                System.out.println( str1 + str2 );
        }
    });

    Thread T_2 = new Thread( new Thread () {
        public void run() {
            synchronized( str2 )
            synchronized( str1 )
                System.out.println( str2 + str1 );
        }
    });

    public static void main( String[] args ) {
        HelloWorldHello hwh = new HellloWorldHello();
        hwh.T_1.start();
        hwh.T_2.start();
    }
}
```

Which statement best describes what happens when the `main` method executes?

A. The program immediately deadlocks.
B. The program runs normally, but the order of Strings printed is unpredictable.
C. The program sometimes prints "HelloWorld" and other times "WorldHello", but sometimes deadlocks.
D. We cannot say because we don’t have enough information in the example.
6. Rewrite the following method by removing the `synchronized` from its methods and replacing each with the equivalent use of `synchronized` in the body of each method.

```java
class Counter {
    private int counter=0;
    synchronized public incr() { counter++ ;}
    synchronized public decr(){ counter--; }
    synchronized public int value() { return counter; }
}
```

Rewrite this class with the required modifications in the space below:
Instructions & Definitions for CMSC 132 Final

• No use of try-catch statements is allowed in any code written in response to any question on this exam;

• For recursive methods: Use only the structures provided in the References provided in this document. Do not use any external data-structures, such as ArrayLists, Stacks, etc. with the intent of removing recursion from any recursive method that you write.

• No question that requires a recursive implementation will allow the explicit use of any iterative statement.

• No auxiliary or helper methods should be required to respond to any of the programming questions on this exam.

• Graph searching questions assume that the natural ordering of vertices will be used in the expansion of adjacencies. In the case of numbers, assume 0, 1, 2, ..., and in the case of literals or Strings, assume the normal rules of English, a, b, c, ....

Allowed definitions for data-structures used on this exam.

Linked Lists

Use the following definition for questions regarding linked-lists. Note: All of these methods are static, accepting SLList<T>s and elements of any object type, T.

public static <T> boolean isEmpty( SLList< T > lst );
public static <T> T first( SLList< T > lst );
public static <T> SLList< T > rest( SLList< T > lst );
public static <T> SLList< T > cons( T item, SLList< T > lst );

<table>
<thead>
<tr>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean isEmpty( lst )</td>
<td>Returns true if lst is empty.</td>
</tr>
<tr>
<td>T first( lst )</td>
<td>Returns the value of the first element in lst. Note, it is an error to call this function on an empty list.</td>
</tr>
<tr>
<td>LinkedList rest( lst )</td>
<td>Returns the “tail,” a SLList of the elements after the first element in a linked list. Note, calling this function on an empty lists causes an error.</td>
</tr>
<tr>
<td>LinkedList cons( ele, lst)</td>
<td>Returns a new SLList that is the result of adding the ele as its first element; elements are kept in their original order.</td>
</tr>
</tbody>
</table>

Table 1: Linked List Operations: All linked list operators are transparent, returning copies.
Binary Trees

Use the following definition for questions regarding Binary Trees.

public class BTree< T > {
    public boolean isEmpty();
    public T getValue();
    public boolean isLeaf();
    public BTree< T > getLeft();
    public BTree< T > getRight();
}

<table>
<thead>
<tr>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean isEmpty()</td>
<td>Returns true if this BTree is empty.</td>
</tr>
<tr>
<td>T getValue()</td>
<td>Returns the value on this BTree object; note, an error results if this tree is empty.</td>
</tr>
<tr>
<td>boolean isLeaf()</td>
<td>Returns true only if this tree is a “leaf,” has empty left and right subtrees. Note: an error results if this method is called on an empty tree.</td>
</tr>
<tr>
<td>BTree&lt;T&gt; getLeft()</td>
<td>Returns the left child. Note: calling this on an empty tree throws an error.</td>
</tr>
<tr>
<td>BTree&lt;T&gt; getRight()</td>
<td>Returns the right child. Note: calling this on an empty tree throws an error.</td>
</tr>
</tbody>
</table>

Table 2: Binary Tree Operations

Ordering Relations

Unless told otherwise, assume that any references to Binary Search Trees are ordered such that their left contains elements less than or equal to the root, and the right contains elements greater than the root. All heaps are Max-Heaps, unless told otherwise.

Provided methods, classes, etc.

Assume that all methods work as described when answering any questions regarding Linked-Lists or Binary Trees on this exam. Do not assume that any other methods exist unless you defined them, using these methods. Do not add any properties to these class definitions.