Final (Scanned)

Monday, December 19, 2016 6:07 PM

he emphasizes trees

seaveled
EE
Data structures vs Database

CS 46B
Fall 2016
Final Exam

Zero points for any answer where you call a method on a primitive.
Only submit your green book. Don't submit this paper.

1) Finding a bug at _____ time is always better than finding it at _____ time. (4 points)
2) What is a hash code collision? (4 points)
3) When does Java create an invisible no-args constructor for a class? (5 points)
4) Define tree, binary tree, and binary search tree. (9 points)
5) Why did this course emphasize MergeSort but not SelectionSort or InsertionSort? Explain your answer in terms of the big-O performance of the algorithms. (3 points).
6) The following method won't compile, because createNewFile() declares that it throws IOException. Rewrite the method 2 different ways so that it compiles, using the 2 different options you saw in class. (4 points)

```
void makeLogFile() {
  File f = new File("log.txt");
  f.createNewFile();
}
```

7) Fill in each blank with 0, 1, or n, where n means any non-negative number (6 points).

   A. In a tree, there are ____ edges into the root node, and ___ edges from the root node.
   B. In a tree, there are ____ edges into an internal node, and ___ edges from an internal node.
   C. In a tree, there are ____ edges into a leaf node, and ___ edges from a leaf node.

8) Why is the following code bad? Assume the type of `galaxy` is `ArrayList<Star>`. (3 points).

```
for (Star s: galaxy)
  if (s.isblack())
    galaxy.remove(s);
```

9) Copy the code below into your green book. Circle all literal strings. (3 points)

```
String s1 = "Planet";
String s2 = "Planet";
String s3 = s2;
String s4 = "Jupiter " + s3;
```

10) What does the following code print? (4 points)

```
String s1 = "ABCDE";
String s2 = "ABCDE";
if (s1 == s2)
  System.out.println("Shallow: yes");
else
  System.out.println("Shallow: no");
if (s1.equals(s2))
  System.out.println("Deep: yes");
else
  System.out.println("Deep: no");
```

11) Implement hashCode(), equals(), and compareTo() methods for the following class. For compareTo(), comparison should be first by maxDepthMeters, then by nAdjacentContinents if necessary, then by name if necessary. (15 points)

```
public class Ocean {
    private String      name;
    private float       maxDepthMeters;
    private int         nAdjacentContinents;
}
```

12) Given the following classes:

```
public class Node<T> {
    private T           data;
    private Node<T>     next;

    public Node(T data)         { this.data = data; }
    public Node<T> getNext()    { return next; }
}

public class LinkedList<T> {
    private Node<T> head;
    private Node<T> tail;

    // Returns the number of nodes in this list whose data
    // is deep-equal to countMe.
    public int nMatching(T countMe) { ??? }

    // Returns true if this list contains a loop.
    public boolean containsLoop() { ??? }
}
```

Write the nMatching() (10 points) and containsLoop() (12 points) methods for the LinkedList class, as described by the comments.

13) Complete the following method (12 points):

```
public void printOceanByMaxDepth(String[] oceanNames,
                                 int[] maxDepthsMeters) { … }
```

Assume names and maximum depths of all oceans are unique, and `maxDepthsMeters[i]` contains the maximum depth of `oceanNames[i]` for any i. Your method should first use an assertion to make sure the two input arrays have the same length. Then print one line for each ocean in `oceanNames`, like the following:

```
Maximum depth of Indian Ocean is 8047 meters.
```

The output should be in ascending order by depth. ZERO POINTS FOR ANY SOLUTION THAT RUNS IN $O(n^2)$ TIME OR WORSE.

14) Complete the `isLeaf` (5 points) and `recurseCountLeavesInSubtreeTreeNode` (10 points) methods in the class below.

```
public class TreeNode<T> {
    private T                       data;
    private HashSet<TreeNode<T>>    children;

    // Returns true if this node is a leaf node.
    public boolean isLeaf()  { ?? Finish this ?? }

    // Returns the number of leaf nodes in the subtree at this node.
    // Returns 1 if this node is a leaf, otherwise returns sum of leaf node
    // counts for all children.
    public int recurseCountLeavesInSubtree() {
        if (isLeaf())
            return 1;
        else {
            ?? Finish this ??
        }
    }
}
```

15) Were you tempted to cheat on this exam? 1 point for any answer.
   A) Yes.
   B) No.
   C) I don't want to say.