

CSE 143 Final Exam

Part 2 - August 19, 2011

Name _____ Student ID # _____

Section _____ TA Name _____

The exam is closed book, closed notes, closed devices, except that you may have a 5x8 card with hand-written notes plus the reference sheet handed out with the exam. There are two blank pages at the end of the exam if you need extra scratch space to write.

You must show your ID and hand in your exam as you leave the room. You may not come back inside after that.

Style and indenting matter, within limits. We're not overly picky about details like an extra or a missing parenthesis, but we do need to be able to follow your code and understand it. A few well-chosen comments can be very helpful. Too many make it hard to read the code.

If you have questions during the exam, raise your hand and someone will come to you. *Don't* leave your seat.

Please wait to turn the page until everyone has their exam and you have been told to begin.

Advice: The solutions to many of the problems are quite short. Don't be alarmed if there is a lot more room on the page than you actually need for your answer.

More gratuitous advice: Be sure to get to all the questions. If you find you are spending a lot of time on a question, move on and try other ones, then come back to the question that was taking the time.

7 - Inheritance	/ 21
8 - Lists	/ 24
9 - Decoding tree	/ 24
10 - Sorting	/ 15
11 - Bonus	/ 1
Total	/ 85

CSE 143 Final Part 2, August 19, 2011

Question 7. (21 points) Inheritance. Consider the following class definitions:

```
public class Insect {
    public void noise() { System.out.println("scratch"); }
    public void hello() { System.out.println("hi"); }
}

public class Ant extends Insect {
    public void noise() { System.out.println("dig dig"); }
}

public class Bug extends Insect {
    public void fly() { System.out.println("flap flap"); }
    public void noise() { System.out.println("Bug says " + what()); }
    public String what() { return "buzz"; }
}

public class Fly extends Bug {
    public void hello() { System.out.println("howdy"); }
    public void fly() {
        System.out.println("fly");
        super.fly();
    }
}

public class Bee extends Bug {
    public void sting() { System.out.println("zap!"); }
    public String what() { return "honey"; }
}
```

Answer the questions about these classes on the next page. You may remove this page from the exam for reference if you wish.

CSE 143 Final Part 2, August 19, 2011

Question 7. (cont.) For each of the following code sequences write down the output produced, if any, if the code executes successfully. If there is either a compile-time or run-time error that prevents successful execution, give a concise description of the problem. If there are errors, they're not trivial ones like incorrect punctuation.

(a) `Insect em = new Ant();`
`em.noise();`

(b) `Bug black = new Fly();`
`black.fly();`

(c) `Bug bunny = new Bug();`
`bunny.noise();`

(d) `Bee b = new Bee();`
`b.sting();`

(e) `Fly house = new Bug();`
`house.fly();`

(f) `Insect thing = new Bug();`
`((Bee) thing).sting();`

(g) `Bug bumble = new Bee();`
`bumble.noise();`

CSE 143 Final Part 2, August 19, 2011

Question 8. (24 points) Suppose we have a class that stores a set of integer values with no duplicates. In this problem the values are stored in a linked list, and the values in the list are kept in sorted, ascending order.

Complete the definition of the method `containsAll` on the next page so it returns `true` if all of the items in a second set (`other`) are also contained in the current set. The method should return `false` if some item is found in the other set that is not contained in the current one. For example, suppose the current set contains `{3, 6, 12, 19, 25}`. If set `x` contains the values `{3, 12, 19}`, then `containsAll(x)` should return `true`. If set `y` contains the values `{6, 19, 42}`, then `containsAll(y)` should return `false`. If the other set is empty, then `containsAll` should return `true`, since there are no items in the other (empty) set that are missing from the current one.

You may define additional private methods and simple variables as needed, but you may not create additional collections, nodes, or other objects. You may not modify either of the sets involved.

For full credit, your method should run in no more than $O(n+m)$ time where n and m are the number of items in the two sets.

The first part of the class definition is given below.

```
public class IntListSet {

    // list nodes
    private class IntListNode {
        public int data;           // value in this node
        public IntListNode next;  // next node or null if none
    }

    // list of values in this IntListSet, stored in ascending order
    // with no duplicates. Null if the set is empty.
    private IntListNode items;

    // construct empty IntListSet
    public IntListSet() { items = null; }
```

Complete the `containsAll` method for this class in the space provided on the next page.

Feel free to remove this page from the exam for reference if you wish.

CSE 143 Final Part 2, August 19, 2011

Question 8. (cont). Complete the method definition below (you may add additional methods if you need them).

```
// Return true if every item in the IntListSet other is also
// contained in this IntListSet. Return false if some item found
// in IntListSet other is not found in this IntListSet.
public boolean containsAll(IntListSet other) {

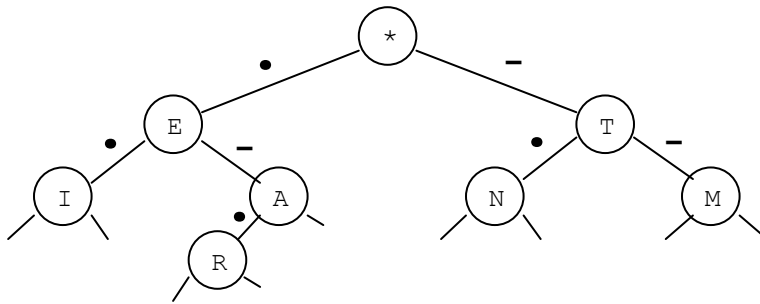
    // write your code below

}
```

CSE 143 Final Part 2, August 19, 2011

Question 9. (24 points) Morse code is a method for encoding letters, digits, and other characters as a sequence of dots and dashes. As with Huffman codes, the codes assigned to frequently used characters are shorter than the codes for infrequent ones. A few examples: the code for **A** is `·-`, **E** is `·` (a single dot), **T** is a single dash `-`, **F** is `··-·`, and **R** is `·-·`.

We can use a binary tree to represent Morse code and to decode individual letters. We will arbitrarily decide that the left subtree of a node is the one reached by a dot (`·`) and the right subtree is reached by a dash (`-`). With those conventions, the top part of the binary tree for Morse code looks like this:



Looking at the tree we can see that a dot represents **E**, the sequence dash dot is **N** (right then left from the root), and dot dash dot is **R** (left, right, left). Unlike Huffman codes the letters may appear anywhere in the tree, not just in the leaf nodes.

For this problem, complete the definition of method `decode` in class `MorseCode` on the next page so that it returns the character whose Morse code is given by the `String` argument `code`. For instance, if the argument `code` is the string `··`, method `decode` should return **I**, which will be the character found in the node reached by starting at the root and following the left (dot) links. If the argument is `·-·`, the result would be **R**.

You may assume that the `code` argument is valid and there is a corresponding node in the tree with that code, and you should assume that the full tree has been initialized properly to contain all of the possible Morse code sequences and characters.

For full credit, your tree traversal must use recursion to descend the tree. You may add additional private methods and simple variables as needed, but you may not create additional collections, tree nodes, or other objects, and you may not modify the tree.

Complete the method `decode` on the next page. You may remove this page for reference while you're working.

CSE 143 Final Part 2, August 19, 2011

Question 9 (cont.) Complete method decode below.

```
public class MorseCode {

    // individual tree node
    private class CodeNode {
        public char ch;           // character in this node
        public CodeNode left;    // subtree reached by a . (dot) from
                                // this node
        public CodeNode right;   // subtree reached by a - (dash) from
                                // this node
    }

    // root of the code tree
    private CodeNode codeTree;

    // constructor
    public MorseCode(...) { ... }

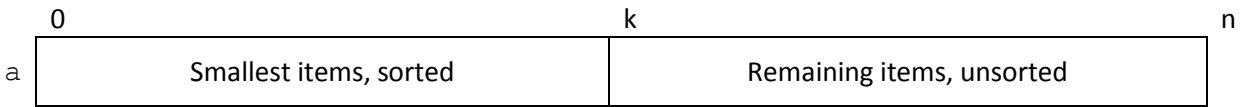
    // return the character corresponding to the string of dots
    // and dashes given in the argument code.
    public char decode(String code) {

        // Write your answer below. Additional space provided on the
        // next page if needed.
    }
}
```

Question 9. (cont.) Additional space for answer if needed.

CSE 143 Final Part 2, August 19, 2011

Question 10. (15 points) Sorting. In lecture we discussed some common sorting algorithms. The *selection sort* algorithm can be described by the following picture of a partially sorted array:



That is, when the sort is partially completed, array elements $a[0..k-1]$ are sorted ($a[0] \leq a[1] \leq \dots \leq a[k-1]$), the remaining array elements $a[k..n-1]$ are unsorted, and all elements in the sorted part $a[0..k-1]$ have values less than or equal to the values in the unsorted part $a[k..n-1]$. Selection sort proceeds by repeating the following steps until no elements remain in the unsorted part:

- Locate the smallest item in the unsorted part $a[k..n-1]$
- Interchange this smallest value with $a[k]$ and increase k by 1

The following code is a partially completed method to perform a selection sort of an integer array. Complete the method by writing the appropriate code in the box. (Note: The problem can be solved without altering any of the given code, so don't change it.)

```
// sort integer array a in non-decreasing order
public void selectionSort(int[] a) {
    for (int k = 0; k < a.length-1; k++) {
```

```
        // Find smallest element in a[k..a.length-1] and swap it with a[k]
```

```
    }
}
```

CSE 143 Final Part 2, August 19, 2011

Question 11. (Bonus – 1 point) “The TAs made me do it”

(All reasonable answers receive the bonus point. All answers that reflect even minimal effort are reasonable. Feel free to use the additional space to include appropriate drawings.)

Your TA has woken up in jail. Explain why.

CSE 143 Final Part 2, August 19, 2011

Additional blank page for scratch paper.

If you write an answer to a test question here, be sure to go back to the original page and write a note so the grader can find your answer.

CSE 143 Final Part 2, August 19, 2011

Additional blank page for scratch paper.

If you write an answer to a test question here, be sure to go back to the original page and write a note so the grader can find your answer.