Name: _____

USC NetID (e.g., *ttrojan*): _____

CS 455 Final Exam
Fall 2015 [Bono]

Dec. 15, 2015

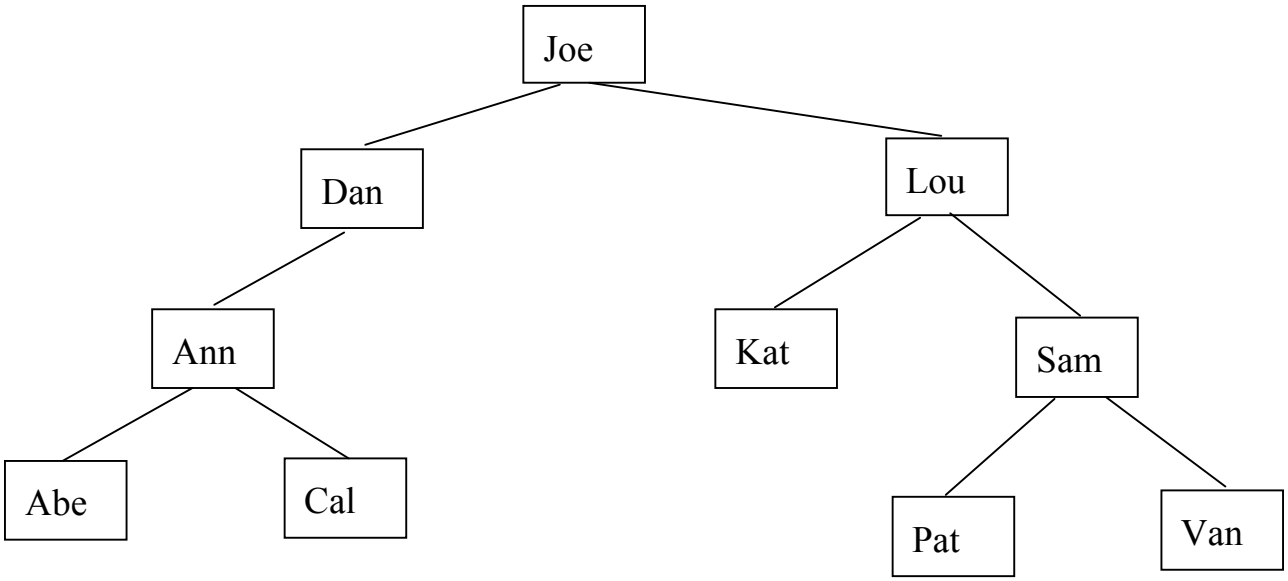
There are 6 problems on the exam, with 70 points total available. There are 10 pages to the exam (5 pages double-sided), including this one; make sure you have all of them. If you need additional space to write any answers, pages 8, 9, and 10 are left blank for that purpose. If you use these pages for answers you just need to direct us to look there.

Note: if you give multiple answers for a problem, we will only grade the first one. Avoid this issue by labeling and **circling** your final answers and crossing out any other answers you changed your mind about (though it's fine if you show your work).

Put your name and USC username (a.k.a., NetID) at the top of the exam. Also put your NetID at the top right of the front side of each page of the exam. Please read over the whole test before beginning. Good luck!

Problem 1 [6 pts. total]

Part A. Consider the following binary search tree, whose root is shown nearest the top of the page (i.e., it's not a sideways tree):



For each of the following lookups from the tree shown above, give the sequence of keys that the target key would have to be compared with to do the lookup.

Part A. lookup *Cal*

Part B. lookup *Molly*

Part C. lookup *Fred*

Problem 2 [2 pts.]

Give the big-O worst case time for checking if an array of size *n* is already sorted.

Problem 3 [10 points]

The following **Java** method is supposed to do what its method comment says below, but it doesn't work correctly in all cases (see the code handout for more about the `Stack` class):

```
/**
   Returns true iff the given string consists of zero or more 'a' characters
   followed by the same number of 'b' characters.
*/
public static boolean sameABs(String str) {

       Stack<Character> s = new Stack<Character>();

       int i = 0;

       while (i < str.length() && str.charAt(i) == 'a') {

          s.push(str.charAt(i));

          i++;

       }

       while (i < str.length() && str.charAt(i) == 'b') {

          s.pop();

          i++;

       }

       return (i == str.length() && s.empty());

    }

}
```

Part A [6]. Come up with some test cases for the function and verify whether they work on the code above. While there are more interesting test cases than just three, **write down three specific test cases as described below:**

A string for which `sameABs` correctly returns `true`:

A string for which `sameABs` correctly returns `false`:

A string for which `sameABs` does not work correctly:

Part B [4]. Fix the code so it works for any string. You should make your modifications directly in the code above, using arrows to show where your new code should be inserted, crossing out code that you would get rid of, etc.

Problem 4 [20 points]

In lecture and a lab we discussed **Java** code to create versions of a concordance that computed the number of occurrences of each distinct word in a file. **Here we're going to make a version of our Concord class that creates a concordance that has all the *locations* of each distinct word in a file**. **We will use the line number as a word's location.**

Here is an example of some input, and the corresponding output when we build a concordance for it and print it (just using `toString`-type format):

```
the big dog went to the big dog
and the other big dog went to the
the big elephant

{and=[2], big=[1, 2, 3], dog=[1, 2], elephant=[3], other=[2], the=[1, 2, 3], to=[1,
2], went=[1, 2]}
```

Note that even if a word appears more than once on a line, we only list that line number once. Some more details of the problem:

- You don't have to worry about putting words into some canonical form for the purposes of this problem (e.g., you may assume there is no punctuation or capital letters in the input).
- **For a file with *n* words your code must build the concordance in at worst *nlogn* time, and be able to print it out in alphabetical order in linear time** (you are not required to write the `toString` code).
- It is expected that you will use the Java library to make your code short and fast. See the code handout for a reminder of some classes and methods. You will be evaluated in part on appropriate use of these tools.
- We have written the code to read the file for you (see next page). **You need to fill in the private data, complete the constructor, and complete `build` by implementing the private method `processLine`.**

Write your answer in the spaces provided on the next page.

Problem 4 (cont.)

```
public class Concord {
   // put private data here:




   public Concord() {        // creates an empty concordance



   }
   public String toString() {  /* you don't have to write this method */ }

   public void build(Scanner in) {  // builds the concordance from data read from "in"
     int currLineNum =  1;
      while (in.hasNextLine()) {
         String currLine = in.nextLine();
         processLine(currLine, currLineNum);  // calls helper function below
         currLineNum++;
      }
   }
   // (helper func) adds all the words in this line to the concordance
   // as occurring on lineNum
   private void processLine(String line, int lineNum) {









   }
}
```

Problem 5 [10 pts. total]

Consider the following **C++** code to dynamically create a `Student` object: (Note: the `Student` class is not shown; you may assume it has a constructor that takes a `String` argument.)

```
Student * joe;              // create a pointer to an object in C++
joe = new Student("Joe");   // dynamically allocate an object in C++
```

Consider each of the four possible headers for a C++ function foo that takes a student as a parameter.

Part A [4]. For each of the headers below, to the right of that header show the corresponding call to pass the student, `joe`, created in the code above, to the function. We started (but didn't finish) the first one for you.

Header	**Corresponding call**
`void foo1(Student s);`	`foo1(`
`void foo2(Student * s);`	
`void foo3(Student * & s);`	
`void foo4(const Student * & s);`	

Part B [2]. In Java all objects are created dynamically (i.e., with new), as was the C++ object created at the top of the page. **Which of the above four function headers and calls most closely corresponds to the semantics of how objects are passed in Java? Write the name of that function below:** [By semantics we mean how it works, not what its syntax is.]

Part C [4]. Draw a box and pointer diagram for the call to the function you indicated in part B. Show all variables used and their values.

Problem 6 [22 points total]

Part A [20]. Write the C++ Boolean function `allUnique`, which takes a linked list of `int`s and returns true iff all the values in the list are unique (i.e., there are no duplicate values in the linked list). Restrictions: the list must be unmodified by the function and the function is only allowed to use a constant amount of extra memory. (In contrast, the `sameABs` problem earlier on the exam used O(*n*) extra memory for a Stack).

The `Node` and `ListType` definitions are on the code handout.

Examples:

list	***return value of call to `allUnique(list)`***
`()`	`true`
`(1)`	`true`
`(3 2 1)`	`true`
`(1 1 1)`	`false`
`(3 2 3)`	`false`
`(4 3 7 3 4 5)`	`false`

```
// PRE: list is a well-formed list.
bool allUnique(ListType list)
```

Part B [2]. What is the worst case big-O time for your solution to part A, as a function of, *n*, the number of elements in ths list:

Extra space for answers or scratch work.

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.

Extra space for answers or scratch work (cont.)

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.

Extra space for answers or scratch work (cont.)

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.