Name: _________________________________________

USC loginid (e.g., ttrojan): __________________________

Midterm Exam 2
CS 455, Spring 2011

March 31, 2011

There are 6 problems on the exam, with 50 points total available. There are 7 pages to the exam, including this one; make sure you have all of them. There is also a separate double-sided one-page code handout. If you need additional space to write any answers, you may use the backs of exam pages (just 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 loginid at the top of the exam. Please read over the whole test before beginning. Good luck!

Remote DEN students only: Do not write on the backs of pages. If additional space is needed, ask proctor for additional blank page(s), put your name on them, and attach them to the exam.

<table>
<thead>
<tr>
<th>Problem</th>
<th>value</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>4 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2 &amp; 3</td>
<td>7 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>8 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 5</td>
<td>16 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 6</td>
<td>15 pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>50 pts.</td>
</tr>
</tbody>
</table>
Problem 1 [4 pts.]
Each of the following hash functions is bad. For each, explain what is wrong with it. Assume that your hash table is an array with indices 0 through HASHSIZE-1. Note: all of the functions return a value in the correct range and do not generate any run-time errors.

Hash function 1.
Assume the hash key is type int.

```java
public static int hash(int key) {
    return 32 * 997 / 53 % HASHSIZE;
}
```

Hash function 2.
Assume the hash key is type String.

Reminder: Random is the class for generating random numbers. `nextInt(n)` returns a random number in the range [0, n-1]

```java
public static int hash(String key) {
    Random rand = new Random();
    int result = 0;
    for (int i = 0; i < key.length(); i++) {
        return result += rand.nextInt(37) * key.charAt(i);
    }
    return result % HASHSIZE;
}
```
Problem 2 [4 pts.]
Consider a Map class to store a collection of $n$ key-value pairs with operations insert, remove, lookup, and printInOrder (the last is to print all entries in order by key). For A-D give worst case unless average case is better.

A. If the Map used an ordered array representation (i.e., ordered by keys), how long would lookup take in big-O terms?

B. How long would printInOrder take if we used an unordered linked list representation?

C. How long would insert take if we used a balanced search tree representation?

D. How long would remove take if we used a hash table representation?

Problem 3 [3 pts.]
Why don’t you need to declare that your method might throw a NullPointerException? How do you normally deal with such exceptions?
Problem 4 [8 points]

Part A. Write code to sort an ArrayList of strings in decreasing order by length. See code handout for more information about the Java sort method. You don’t have to worry about the order of words that have the same length. Hint: you will need an additional class. Here is an example:

words before call: ("a" "sandal" "dog" "label")
words after call to sortDecrLength: ("sandal" "label" "dog" "a")

public static void sortDecrLength(ArrayList<String> words)
Problem 5 [16 pts. total]
Implement a class for a Stack of int’s that uses an array (not ArrayList) representation. To keep the code simpler you don’t need reallocate when you run out of space: instead use a fixed size array of size CAPACITY; the isFull() method allows the client to check if there is space to push any more values. The interface for the class is given below: add the necessary private data and method bodies. All your non-constructor methods must run in O(1) time.

public class Stack {
    private static final int CAPACITY = 100;
    // Reminder: array -- not ArrayList

    // create an empty stack of ints
    Stack() {

    // returns true iff there is no more space to push any more elements
    boolean isFull() {

    // returns true iff the stack has no elements in it
    boolean isEmpty() {

    // class definition continued next page]
Problem 5 (cont.)

// push item onto the top of the stack
// PRE: !isFull()
    void push(int item) {

    }

// remove the top element from the stack
// PRE: !isEmpty()
    void pop() {

    }

// return the top element in the stack (stack is unchanged)
// PRE: !isEmpty()
    int top() {

    }
Problem 6 [15 points]

Write the boolean function `isConsecutive`, which returns true iff its `LinkedList` parameter is a sequence of increasing consecutive integers.

Examples (Note: linked lists shown below as abstract sequences of numbers):

```
list     Return value of `isConsecutive(list)`:
(3 7 5)   false
(3 4 5 6) true
(3 5 7 9) false
(3 3 3)   false
(3 4 5 7) false
(6 5 4 3) false
(-1 0)    true
(3)       true
<empty>   false
```

// returns true iff the list is a sequence of increasing
// consecutive integers.
// Special cases: false for empty list, true for 1-elmt list.
public static boolean isConsecutive(LinkedList<Integer> list) {