Name: ____________________________________________

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

Midterm Exam 2
CS 455, Fall 2012

Nov 7, 2012

There are 7 problems on the exam, with 62 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!

<table>
<thead>
<tr>
<th>Problem</th>
<th>value</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1 &amp; 2</td>
<td>8 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>10 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>4 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 5</td>
<td>10 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 6</td>
<td>10 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 7</td>
<td>20 pts.</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td>62 pts.</td>
</tr>
</tbody>
</table>
Problem 1 [2 pts. total]
Consider a hash table of size $b$ (i.e., the size of the array) that uses chaining that currently holds $n$ map entries. Assume we are using a high quality hash function.

Part A. What is the number of array elements (not map entries) you need to examine to do the map operation contains(key) on the hash table? Circle the answer that best matches:

i. 1

ii. $n$

iii. $b$

iv. $n/b$

v. $b/n$

Part B. What is the number of map keys you need to examine to do the map operation contains(key) on the hash table? List the answer from part A that best matches:

Problem 2 [6 pts. total]
Assume you have an array of names with the following contents (numbers above the line are indices):

<table>
<thead>
<tr>
<th></th>
<th>Ann</th>
<th>Bob</th>
<th>Carly</th>
<th>Dave</th>
<th>Ed</th>
<th>Frida</th>
<th>Hal</th>
<th>Hank</th>
<th>Iris</th>
<th>Joe</th>
<th>Kat</th>
<th>Lou</th>
<th>Sam</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For each of the targets given below, list the names that the target would be compared with in a binary search on the array. List them in the order that the comparisons would be done.

1. target = Carly

2. target = Mary

3. target = Dave
Problem 3 [10 points]

For each of the following time complexity values, name or briefly describe one algorithm we have discussed in this class that has that time complexity (it can be average or worse case time). For each of your answers you need to include what data structure your algorithm is operating on and what \( n \) represents (i.e., what constitutes the size of the problem for this particular problem). **Please limit the length of your answer to the space provided below – each one can be answered in just one sentence.**

---

\( O(1) \)

---

\( O(\log n) \)

---

\( O(n \log n) \)

---

\( O(n) \)

---

\( O(n^2) \)

---

[please do not write below this line unless you had to cross out something above]
Problem 4 [4 points]
Consider the following Java program that uses exceptions (see code handout definition of BadIntException and more on exception classes):

```java
public class MyProg {
    public static void main(String[] args) {
        try {
            Scanner in = new Scanner(System.in);
            System.out.println("Please enter an integer: ");
            String line = in.nextLine();
            Scanner lineScanner = new Scanner(line);
            int val = getInt(lineScanner);
            if (val > 1000) {
                throw new BadIntException("Value out of range");
            }
            System.out.println("Value entered was: " + val);
        }
        catch (BadIntException exc) {
            System.out.println("Bad Int Exception: " + exc.getMessage());
        }
    }
    public static int getInt(Scanner lineScanner) {
        try {
            if (!lineScanner.hasNext()) {
                throw new BadIntException("Problem #1");
            }
            if (!lineScanner.hasNextInt()) {
                throw new BadIntException("Problem #2");
            }
            int val = lineScanner.nextInt();
            if (lineScanner.hasNext()) {
                throw new BadIntException("Problem #3");
            }
            return val;
        }
        catch (IOException exc) {
            System.out.println("IO ERROR: " + exc.getMessage());
        }
        return -100;
    }
}
```

Does MyProg compile? __________
If you answered "yes": make the minimal changes to it above so it would compile.
If you answered "no": what exact output does the current code give if the user enters the following input at the prompt:

7 i
Problem 5 [10 points]

For the purposes of this problem, assume static methods `printStack` and `printQueue` work as described on the code handout.

Show the output of the following function, `prob5Func`, when given the following value for `s` (shown as it would be printed by `printStack`):

top: [5, 4, 2, 7, 6, 1]

```java
public static void prob5Func(Stack<Integer> s) {
    Queue<Integer> tmp = new LinkedList<Integer>();
    int num = 0;
    while (!s.empty()) {
        tmp.add(s.pop());
        num++;
    }
    printQueue(tmp);
    printStack(s);
    for (int i = 0; i < num; i++) {
        int curr = tmp.peek();
        tmp.remove();
        if (curr % 2 == 1) {
            s.push(curr);
        } else {
            tmp.add(curr);
        }
    }
    printQueue(tmp);
    printStack(s);
    while (!tmp.isEmpty()) {
        s.push(tmp.remove());
    }
    printQueue(tmp);
    printStack(s);
}
**Problem 6 [10 points]**

The `Student` class with the interface shown below is used in this problem:

```java
public class Student {
    . . .
    public Student(String name, int score) { . . . }
    public String getName() { . . . }
    public int getScore() { . . . }
    public void setScore(int newScore) { . . . }
}
```

Implement the method specified below. For full credit your code should not create any more objects than necessary.

```java
/**
   * addBonus adds bonusPoints points to the scores of all students in the
   * given classList.
   */
public static void addBonus(ArrayList<Student> classList, int bonusPoints) {
```
Problem 7 [20 points]
Write the function `removeAdjacentEvens`, which removes from a linked list any even numbers that directly follow another even number in the list. Examples:

<table>
<thead>
<tr>
<th>list</th>
<th>list after call to <code>removeAdjacentEvens(list)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>(6 2 5 2 8 4 3)</td>
<td>(6 5 2 3)</td>
</tr>
<tr>
<td>(4 2 5 1)</td>
<td>(4 5 1)</td>
</tr>
<tr>
<td>()</td>
<td>()</td>
</tr>
<tr>
<td>(5 3 7)</td>
<td>(5 3 7)</td>
</tr>
<tr>
<td>(2 6 4)</td>
<td>(2)</td>
</tr>
</tbody>
</table>

// removes from the linked list all even numbers that immediately follow // another even number

```java
public static void removeAdjacentEvens (LinkedList<Integer> list) {
```