Statement of integrity: I did not, and will not, break the rules of academic integrity on this exam:

(Signature)

Circle Your Section:

<table>
<thead>
<tr>
<th>Tuesday</th>
<th>Wednesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>HO 306</td>
<td>PH 307</td>
</tr>
<tr>
<td>HO 401</td>
<td>PH 213</td>
</tr>
<tr>
<td>PH 407</td>
<td>UH 111</td>
</tr>
<tr>
<td>12:20</td>
<td>5: Barr</td>
</tr>
<tr>
<td>1:25</td>
<td>9: Barr</td>
</tr>
<tr>
<td>2:30</td>
<td>8: Swamy</td>
</tr>
<tr>
<td>3:35</td>
<td>8: Swamy</td>
</tr>
<tr>
<td>1: Renaud</td>
<td>6: Renaud</td>
</tr>
<tr>
<td>2: Scovetta</td>
<td></td>
</tr>
<tr>
<td>3: Barr</td>
<td></td>
</tr>
<tr>
<td>4: Ang</td>
<td></td>
</tr>
</tbody>
</table>

Instructions:

- Read all instructions carefully, and read each problem completely before starting it!
- This test is closed book – no calculators, reference sheets, or any other material allowed.
- You must use blue or black pen or pencil.
- Conciseness, clarity, and style all count. Show all work (e.g., algorithms, box diagrams) to receive partial credit.
- Carefully (but briefly) comment each control structure and major variable.
- If you use break (not in switch) or System.exit to exit any control structure, you will lose points!
- You may not use any MATLAB code.
- You may not alter, add, or remove any code that surrounds the blanks and boxes.
- Only one statement, expression, modifier, type, or comment per blank!
- Use the backs of pages if you need more space or scrap. You may request additional sheets from a proctor.
- If you supply multiple answers, we will grade only one.

Core Points:

1. ________ (30 points) _________
2. ________ (20 points) _________
3. ________ (40 points) _________
4. ________ (10 points) _________
Total: ________/ (100 points) _________

Bonus Points:

Bonus: ________/ (3 points)
Java Reminders

This is not a problem. This is a “cheat sheet” called Java Reminders! You might need some of the following methods in this exam.

Class Math
- int max(int x, int y) return the maximum of integers x and y
- int min(int x, int y) return the minimum of integers x and y
- double random() return a double between 0 (inclusive) and 1 (exclusive)

Class Object
- boolean equals(Object arg) return true if current Object is equal to arg
- String toString() return String description of the current Object

Class String
- char charAt(int i) return the character at position i in the current String
- int compareTo(String s) compare current String with s. Return 0 (zero) if identical
- boolean equals(String s) return true if current String has same contents as s
- int indexOf(char c) return the position of first occurrence of c in the current String
- int length() return the length of a String
- new String(char[] c) return a reference to a new String with the same contents as character array c
- char[] toCharArray() create a character array with the same contents of the current String

Class TokenReader
- int readInt() read an integer from user input
- String readString() read a word with no whitespace from user input: use for reading Strings
Problem 1  [30 points] Characters & Strings, Searching

Class Data is used by Problem 2 to create an array of Data objects. Each object will have, as its instance variables, a random String (s) and a count of the most repeated character in the String (max). In this problem, you must complete the following methods that set s and max:

- **setString**: create a String using an array of n random characters between ‘a’ and ‘z’. The result is stored in s.
- **setMaxCount**: find the count of the most repeated character in String s. The maximum count is stored in max. For instance, given "babdbbacdb", setMaxCount would assign 6 to max.

Hints: Look on page 2 for useful methods, like charAt(int i) and Math.random().

```java
public class Data {
    private String s; // random String set by setString
    private int max; // count of most repeated character in $s$
    public Data(int n) { setString(n); setMaxCount(); } // constructor
    public String getString() { return s; }
    public int getMaxCount() { return max; }
    public String toString() { return "String: "+s+"; Max count: "+max; }

    // Generate a String of $n$ random characters between ‘a’ and ‘z’. Store the
    // result in instance variable $s$:
    private void setString(int n) {
        char[] tmp = new char[n];
        for(int i=0; i<n; i++)
            tmp[i] = (char) (Math.random()*(‘z’-‘a’+1)+‘a’);
        s = new String(tmp);
    }

    // Find the count of the most repeated character in $s$ and store in $max$:
    private void setMaxCount() {
        max = 1; // max so far
        for (int i=0; i < s.length(); i++) {
            char tmpchar = s.charAt(i);
            int tmpsum = 1;
            for (int j = i+1; j < s.length(); j++ )
                if (tmpchar == s.charAt(j))
                    tmpsum++;
            max = Math.max(tmpsum,max);
        }
    }
}
```
Problem 2  [20 points] Arrays of objects, Sorting

Assume that you successfully completed the methods in Problem 1 for class Data. Problem 2 uses class Data to create an array of Data objects. Each object’s String s is set to a random amount of characters between 1 and 25, inclusive. In main below, the array d collects these objects and is sorted by the sortData method. You must use either a select or insert sort technique to complete sortData. Hint: sortData does not create a new array!

```java
public class Problem2 {
    // Create and sort an array of Data objects:
    public static void main(String[] args) {
        // Create an array of $SIZE$ Data objects with Strings of random length between 1 and 25:
        final int SIZE=5, LMIN=1, LMAX=25;
        Data[] d = new Data[SIZE];
        for (int i = 0 ; i < SIZE ; i++)
            d[i] = new Data( (int) (Math.random()*(LMAX-LMIN+1)+LMIN) );
        // Sort the array $d$:
        sortData(d);
    } // Method main

    // Sort the array of Data objects $d$ according to each count of the most repeated character in descending order:
    private static void sortData(Data[] d) {

        // using select sort (descending order)
        int index, minIndex, scanIndex;
        Data tmp;

        // look at each element, start L->R
        for(index=0; index<d.length-1; index++) {
            minIndex = index;

            // inner loop finds smallest value
            for(scanIndex = index+1; scanIndex < d.length; scanIndex++)
                if (d[scanIndex].getMaxCount() > d[minIndex].getMaxCount())
                    minIndex = scanIndex;

            // swap values
            tmp = d[minIndex];
            d[minIndex] = d[index];
            d[index] = tmp;
        }
    } // Method sortData
}
```

} // Class Problem2
Problem 3 [40 points] Multidimensional arrays, Carpenter Lab

Background: Suppose we want to study Carpenter Lab’s use per shift per day per week. We shall use a 3D array $t$ to store the number of questions each consultant gets in a 1-hour shift: 1st dimension=day of week, 2nd dimension=time of shift, 3rd dimension=week of data. All Mon & Wed shifts record for 2 weeks, all Tue & Fri shifts record for 1 week, and all Thu shifts record for 3 weeks. For example, $t[0][1][1]$ is the number of question recorded on Mon, 2nd shift, and 2nd week.

Problem: Complete class Problem3 to build the 3D array $t$ that will eventually store the counts of questions:

- **startTime** and **stopTime**: arrays that store the first and last shift times for each day. The times are given in military hours. For example, **startTime[0]** is the starting time (13) of the first shift on Mon. **stopTime[0]** is the ending time (15) of the last shift on Mon. Hint: So, there are two shifts on Mon (13–14 and 14–15). Note: We use a 5 day week.
- **weeks**: the number of weeks that data is recorded for a particular day. For example, **weeks[1]** is the number of weeks the Tue shifts record data.
- **main**: declares $t$. Note: We’re only creating the array to hold the responses, not obtain them.
- **createTable**: fills in the rest of $t$. The first index of $t$ corresponds to all shifts on a day. For example, $t[0]$ refers to an array containing all Mon shifts. The second index corresponds to the weeks recorded for each shift on that day. For example, $t[0][0]$ refers to an array containing the data for the first shift on Mon, $t[0][1]$ corresponds to the second shift on Mon, and so forth.
- **printTable**: prints the default values stored inside $t$ for all weeks. Print a blank space where there is no shift and/or data. In the portion of the output below, note how Tue on Week 2 has no data because Tue shifts only record for 1 week. Hint: We used **maxArray** to find maximum array values that you need to use in your loops, but you shouldn’t use it.

Portion of output:

```
Week  1
M T W R F
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

Week  2
M T W R F
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

```java
public class Problem3 {
    public static int[] startTime = {13,13,13,13,13}; // starting times for the shifts
    public static int[] stopTime = {15,17,17,17,16}; // stopping times for the shifts
    public static int[] weeks = { 2, 1, 2, 3, 1}; // weeks data is collected each day
    public final static int DAYS = startTime.length;
    public final static int MAXHOURS = maxArray(stopTime);
    public final static int COUNTMAX = maxArray(weeks);
    public final static int MAXDIFF = maxArray(stopTime)-maxArray(startTime);

    // Create and print initial table:
    public static void main(String[] args) {
        int[][][] t = createTable(); // create initial array $t$
        printTable(t); // print current state of table $t$
    } // method main

    // Find maximum value in a 1D array:
    private static int maxArray(int[] x) {
        int max = x[0];
        for(int i=1;i<x.length;i++) max = Math.max(max,x[i]);
        return max;
    } // method maxArray
```
// Return a 3D array that to hold number of responses for all
// the days of the week, the shifts on each day, and the weeks:
private static int[][][] createTable() {

    int[][][] temp = new int[DAYS][][]; // create temp 3D array
    for (int col=0; col < DAYS; col++) {
        temp[col] = new int[stopTime[col]-startTime[col]][];
        for (int row=0; row < MAXDIFF; row++)
            if (row < stopTime[col]-startTime[col])
                temp[col][row] = new int[records[col]];
    }
    return temp;
}

// Output the 3D array $t$. Print a blank space where there is no data:
private static void printTable(int[][][] t) {

    for (int d = 0; d < COUNTMAX; d++) {
        System.out.println("Week "+(d+1));
        System.out.println("M T W R F");
        for(int row=0; row < MAXDIFF; row++) {
            for (int col=0; col < DAYS; col++) {
                if (row < stopTime[col]-startTime[col] && d < records[col])
                    System.out.print(t[col][row][d] + " ");
                else
                    System.out.print(" ");
            }
            System.out.println();
        }
        System.out.println();
    }
}

} // class Problem3
Problem 4  [10 points] Inheritance of public members, constructor chaining

The following program prints 5 lines of output which you must determine. Hint: There is only one number per blank.

```java
public class Problem4 {
    public static void main(String[] args) {
        B b = new B();
        A a = b;
        a.method1(b.x+b.y);
        System.out.println(a.x);
    }
}

class A {
    public int x;
    public A(int x) {
        method1(x);
        this.x = x;
    }
    public void method1(int x) { System.out.println(x); }
    public int method2() { return x; }
}

class B extends A {
    public int x = 1;
    public int y = 2;
    public B() {
        this(4);
        System.out.println(y); }
    public B(int x) {
        super(x);
        System.out.println(method2());
    }
    public int method2() {
        return y+super.method2();
    }
}

Output:
1) 4
2) 6
3) 2
4) 3
5) 4
```
Checklist: Congratulations! You reached the last page of Prelim 3. Make sure your name, ID, and section are clearly indicated. Also, re-read all problem descriptions/code comments/instructions. If you reached this part before exhausting the allotted time, check your test! Maybe you made a simple mistake? Please check the following:

___ maintained all assumptions
___ remembered semicolons
___ didn’t confuse equals with assign operators
___ completed all tasks
___ filled in ALL required blanks
___ given comments when necessary
___ declared all variables
___ maintained case-sensitivity
___ handled “special cases” correctly
___ indicated which solution to grade if you wrote multiple attempts

Bonus: [3 points] Remember that bonus points do not count towards your core-point total! You will lose additional points from your entire CS100J bonus score for “inappropriate” language. To receive bonus points, tear this sheet off from the exam, make sure the proctor records the points on the front page, and put it in a separate pile to maintain anonymity.

On a scale of 0 (being worst) to 5 (being great), please rate how well the following course materials helped you learn if you used them. Otherwise, mark the ones you didn’t use with NA (not applicable):

1) _______ Savitch
2) _______ ProgramLive
3) _______ Lewis & Loftus
4) _______ Java In a Nutshell
5) _______ Online lecture notes
6) _______ Online examples
7) _______ Solutions to projects, exercises, prelims
8) _______ Section Notes (Indicate which TA: ________________________)

If you frequently used Savitch, I would like to know what you really think of it. Do you feel strongly about keeping or ditching Savitch? Please explain why: