
(Print last name, first name, middle initial/name)

(Student ID)

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

(Signature)

Circle Your Section:

	Tuesday			Wednesday	
	HO 306	HO 401	PH 407	PH 213	UH 111
12:20	5: Barr				
1:25	1: Renaud	2: Scovetta		6: Renaud	
2:30		3: Barr			8: Swamy
3:35		4: Barr			8: Swamy

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.
- Conciseness, clarity, and style all count. Show all work to receive partial credit.
- Carefully comment each control structure and major variable.
- If *you* use **break** or **System.exit** to exit any control structure (except **switch**), you will lose points!
- You may **not** use Java arrays or 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. _____ (18 points) _____
 2. _____ (32 points) _____
 3. _____ (50 points) _____
 Total: _____ / (100 points) _____

Bonus Points:

Bonus: _____ / (3 points)

Problem 1 [18 points] *Short-answer: course policies, fundamental concepts, literals, operators, expression statements*

Ia [5 points] Fill in the following blanks with the appropriate words:

Programming is _____ problem solving.

MATLAB stands for _____ .

Java is strongly _____ .

David I. Schwartz's office is in 513__ Upson Hall.

Ib [6 points] Fill in the blanks for the following code fragment that swaps the contents of **a** with **b** without directly assigning 1 to **b** and 2 to **a**. Hint: Use **tmp**:

```
int a=1, b=2, tmp;
```

```
_____ = _____ ; _____ = _____ ; _____ = _____ ;
```

Ic [1 point] Fill in the blank for the output for the following statement:

```
System.out.println("Output c: " + ( true && ( false || !( true || false ) ) ) );
```

Output c: _____

Id [1 point] Fill in the blank for the output for the following statement:

```
System.out.println("Output d: " + (1-2-3-4) );
```

Output d: _____

Ie [1 point] Show the output for the following statement:

```
System.out.println("Hi\nBye!");
```

If [2 points] Fill in the blanks for the output for the following statements:

```
int x=2, y=1;
```

```
x = y-- -x;
```

```
System.out.println("x: "+x+" y: "+y);
```

x: _____ y: _____

Problem 2 [38 points] *Selection statements, remainder operator, Strings, relations, user I/O*

Complete the following code in class **Problem2** by filling in the blanks and boxes. The user is prompted to enter a **temperature**, which is rated using strings according to the following criteria:

temperature	rating	temperature	rating	temperature	rating
90 – 92	"almost hot"	93 – 96	"hot"	97 – 99	"very hot"
80 – 82	"almost warm"	83 – 86	"warm"	87 – 89	"very warm"
70 – 72	"OK"	73 – 76	"OK"	77 – 79	"OK"
60 – 62	"very cool"	63 – 66	"cool"	67 – 69	"almost cool"
50 – 52	"very cold"	53 – 56	"cold"	57 – 59	"almost cold"

To determine the **rating**, your program will:

- obtain a user input **temperature**. Assume that the user enters a value between 50 and 99, inclusive. Do not check for illegal input.
- determine an initial **rating** based on initial temperature ranges 90 – 99, 80 – 89, 70 – 79, 60 – 69, and 50 – 59.
- modify **rating** based on the specific temperature range, as demonstrated in the chart above. Note that the **rating** modifications of "warm" and "hot" are different from the modifications of "cool" and "cold". For instance, given a **temperature** of 81, the **rating** is "almost warm", but for a **temperature** of 51, the **rating** is "very cold". Hint: A remainder operation (using %) can help identify the specific portion of the initial range.

Examples of different sessions:

```
Enter temperature: 52
Your temperature is very cold.
Enter temperature: 67
Your temperature is almost cool.
Enter temperature: 75
Your temperature is OK.
Enter temperature: 83
Your temperature is warm.
Enter temperature: 99
Your temperature is very hot.
```

```
public class Problem2 {
    public static void main(String[] args) {

        // Initialize variables. Assume temperature entered between [50,99]:
        TokenReader in = new TokenReader(System.in); // input reader
        System.out.print("Enter temperature: "); // prompt for temperature
        double temperature = in.readDouble(); // user-input temperature
        String rating; // temperature rating

        // Apply initial temperature rating:

        if ( temperature >= ____ ) rating = " _____ ";

        else if ( temperature >= ____ ) rating = " _____ ";

        else if ( temperature >= ____ ) rating = " _____ ";

        else if ( temperature >= ____ ) rating = " _____ ";

        else rating = " _____ ";
    }
}
```

```
// Modify $rating$ based on the specific portion of the temperature range:
```

```
// Output the modified temperature rating:
```

```
System.out.println("Your temperature is " + _____ + ".");
```

```
} // method main
```

```
} // class Problem2
```

Problem 3 [50 points] *Algorithms, Repetition: conditional update, accumulation*

Background: *DIS.com* needs to ship boxes to consumers but is too cheap to hire more than one worker. A random amount of boxes (1 to 4, inclusive) spews forth from a chute into a bin. The lonely worker must take boxes out of this bin and place them in a truck for shipping. Because of sporadic back pains, the worker will take a random amount of boxes (between 1 and 4, inclusive) out of the bin. Unfortunately, the bin may hold a maximum of 7 boxes. To prevent the bin from filling up and shutting down the whole operation, the worker must keep working. Although the worker starts completely refreshed, all this work tires the worker. So, every four trips between the bin and truck the worker's efficiency drops by 25% of the previous value. Eventually the worker will cease carrying enough boxes, and thus, the bin will fill up.

Algorithm: You simulate this problem with a program that has this algorithm:

- Set up initial values.
- Add boxes to bin.
- If the bin is not full:
 - remove boxes that the worker is able to extract.
 - obtain more boxes from the chute.
 - repeat.
- Otherwise, stop the simulation and report the results.

Tasks: Fill in the blanks below to complete the code that performs the simulation:

- Initialize variables that represent in these initial amounts: boxes taken from the bin (**boxesTaken**), boxes inside the bin (**boxesInBin**), and total amount boxes extracted by the worker (**totalBoxes**).
- Use a loop to perform each cycle of the algorithm. Each cycle consists of checking if the bin is full, updating the count of cycles and amounts of boxes, decreasing worker efficiency (if necessary), and obtaining a new batch of boxes to add to the bin.
- Report the count of cycles of the simulation and the total amount of boxes taken by the worker.

Notes: Remember that you may *not* use arrays. You must use the code and blanks supplied for you!

```
public class Problem3 {

    public static void main(String[] args) {

        // Initialize variables:

        int MIN      = 1;    // minimum number of boxes for chute and worker

        int MAX      = 4;    // maximum number of boxes for chute and worker

        int MAXBOXES = 7;    // maximum number of boxes that the bin can hold

        // random amount of boxes (MIN to MAX, inclusive) added to bin:

        int boxesAdded =
            _____ ;

        int boxesInBin = _____ ; // current # of boxes in bin

        int boxesTaken = _____ ; // current # of boxes taken from the bin

        int totalBoxes = _____ ; // total # of boxes taken so far

        double eff     = _____ ; // worker efficiency, which starts at 100%

        int count      = _____ ; // count of cycles so far
    }
}
```

```
// Process each cycle of the simulation:
while ( _____ ) {
    _____ ; // increment # of cycles
    _____ ; // increment # of boxes in bin

    // Determine current # of boxes taken from bin. The worker may not
    // extract more boxes than are already in the bin:
    boxesTaken =
        _____ ;

    if ( _____ <= _____ ) {
        _____ ;
        _____ ;
    } else {
        _____ ;
        _____ ;
    }

    // Reduce worker's efficiency by 25% every 4th cycle of the simulation:
    if ( _____ )
        _____ ;

    // Obtain new boxes to attempt putting in bin for the next cycle:
    _____ ;
}

// Report results:
System.out.println("Total boxes taken:\t" + _____ );
System.out.println("Number of cycles of simulation:\t" + _____ );

} // method main
} // class Problem3
```

Checklist: Congratulations! You reached the last page of Prelim 1. 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.

1) What are 1 to 3 things we can do to improve lecture? (You may also say what you like, as well.)

2) What are 1 to 3 things we can do to improve section? (You may also say what you like, as well.)

3) What are 1 to 3 things we can do to improve CS100J, overall? (You may also say what you like, as well.)