CS312 Fall 2019 Exam 1 Solution and Grading Criteria.
Grading acronyms:
BOD - Benefit of the Doubt. Not certain code works, but, can't prove otherwise
Gacky or Gack - Code very hard to understand even though it works. (Solution is not elegant.)
LE - Logic error in code.
NAP - No answer provided. No answer given on test
NN - Not necessary. Code is unneeded. Generally, no points off
OBOE - Off by one error. Calculation is off by one.
RTQ - Read the question. Violated restrictions or made incorrect assumption.

1. Expressions: 1 point each, 16 points total

A.	`6 * 2 - 3 * (5 - 10)`	**`27`**	
B.	`10 / 4 * 20 / 15`	**`2`**	
C.	`0.5 + 5 / 2 * 2.5`	**`5.5`**	
D.	`100 / (20 * 20) * 703.0`	**`0.0`**	
E.	`50 % 15 + 33 % (10 * 6)`	**`38`**	
F.	`5.125 % 0.25`	**`0.125`**	
G.	`10 * -5 + 100 + "GDC"`	**`"50GDC"`**	quotes required
H.	`8 - 2 + "8-2" + 8 * 2 + 8`	**`"68-2168"`**	quotes required
I.	`"" + 8 % 16 * 10 + 2`	**`"802"`**	quotes required
J.	`(int) (-1.9 * 2)`	**`-3`**	
K.	`(int) 3.75 * 3 + 13 % 5`	**`12`**	
L.	`(double) (10 / 4 * 3)`	**`6.0`**	
M.	`"Java" + 3 * 4`	**`"Java12"`**	quotes required
N.	`Math.pow(2.0, 3.0)`	**`8.0`**	
O.	`Math.floor(100.5 / 30.5)`	**`3.0`**	
P.	`(int) Math.random() * 4 + 2`	**`2`**	(int) cast higher precedence than *

2. Code tracing: 2 points each, 24 points total. Place you answer in the box to the right of the code. If the code results in a syntax error, answer **syntax error.** If the code results in a runtime error, answer **runtime error.**

AS SHOWN or - 2. First two instances of "answer" counted wrong.

A. `12 7`
B. `30.0 2.0`
C. `70`
D. `25`
E. `190`
F. `-10 2 10 -8 -5 5`
G. `CS312 S3`
H. `\12\12`
I. `-18-9-17 -1-8`
J. `16 3`
K. `-2.0 3.0 -5.0`
L. `96`

3. Conditionals: 12 points:

```
    public static String getSemester(int month, int day) {
        String result = "Fall";
        if ((month == 1 && day >= 15)
                || (1 < month &&  month < 5)
                || (month == 5 && day <= 20)) {
            result = "Spring";
        } else if ((month == 5 && day >= 21)
                || (5 < month && month < 8)
                || (month == 8 && day <= 25)) {
            result = "Summer";
        }
        return result;
}
```

```
Somewhat shorter test
      String result = "Summer";
      if ((month == 8 && day >= 26)
              || (month > 8)
              || month == 1 && day <= 14) {
          result = "Fall";
      } else if (month < 5 || month == 5 && day <= 20) {
          result = "Spring";
      }
      return result;
```

```
Alternate Solution
        // simpler? Convert to number 101 for January 1 to 1231 for December 31
        int totalDays = month * 100 + day
        String result = "Fall";
        if (115 <= totalDays && totalDays <= 520) {
            result = "Spring";
        } else if (521 <= totalDays && totalDays <= 825) {
            result = "Summer";
        }
        return result;
```

consider 3 options. In other words, attempt to address the three possible outcomes, 4 points
correct use of &&, ||, ==, or multiple ifs, 3 points
correct logic for special cases, splits between months, 4 points
return String result, 1 point

Other:
not using <= and or >= , -2 (for example > underlined)

4. Programming: 16 points

```
public static void countTensPlace(Scanner key) {
    System.out.print("Ten's place digit to count: ");
    int target = key.nextInt();
    System.out.println();
    System.out.print("How many numbers? ");
    int numValues = key.nextInt();
    System.out.println();
    int count = 0;
    for (int i = 1; i <= numValues; i++) {
        System.out.print("Enter " + i + ": ");
        int tensDigit = (key.nextInt() / 10) % 10;
        if (tensDigit == target || tensDigit == -target) {
            count++;
        }
    }
    double percent = 100.0 * count / numValues;
    System.out.println();
    System.out.println(percent + "% values ended with " + target
            + " in the ten's place");
}
```

- **prompt and read target int, 1 point (call to nextLine okay, but not needed. Only reading ints)**
- **prompt and read in number of values, 1 point**
- **cumulative sum variable outside loop and initialized to 0 to count number of successes, 1 point**
- **loop with proper bounds, 3 points**
- **prompt and read in current value in loop, 1 point**
- **correctly extract ten's place digit, 3 points**
- **correctly check if ten's place matches target (no need to check negative), 2 points**
- **increment cumulative sum variable correctly, 1 point**
- **calculate percentage correctly, 2 points (must be correct double calculation)**
- **print out result correctly, 1 point**

Other deductions:

Math methods: -3
Using Strings, especially if trivializes extracting 10's place digit, -6

5. Programming, Strings: 12 points

```java
public static String replace(String source, char tgt, String replace) {
    String result = "";
    for (int i = 0; i < source.length(); i++) {
       char currentChar = source.charAt(i);
       if (currentChar == tgt) {
           result += replace;
       } else {
           result += currentChar;
        }
    }
    return result;
}
```

- create empty resulting String, 1 point
- loop through all characters of source, 4 points (-1 if do not call length method correctly)
- use charAt to get current char, 2 points
- check if current char equals target char correctly, 2 points
- concatenate replacement or current char correctly, 2 points
- return result, 1 point

Other:
non allowed String methods, -3 to -10 depending on severity
no () on method calls, -1
new as a keyword, -1
infinite loop, -4
early return, -3
cast char to String, -1

6. Graphics Programming: 20 points

```
    public static void drawFigure(Graphics g, int x, int y, int size,
            int numRectangles) {

        g.setColor(Color.BLUE);
        g.fillRect(x, y, size, size);
        int xOffset = 0;
        int rectWidth = size / numRectangles;
        int rectHeightIncrease = rectWidth * 2;
        int rectHeight = rectHeightIncrease;
        // draw the left side rectangles
        for (int i = 0; i < numRectangles / 2; i++) {
            if (i % 2 == 0) {
                g.setColor(Color.GRAY);
            } else {
                g.setColor(Color.WHITE);
            }
            g.fillRect(x + xOffset, y, rectWidth, rectHeight);
            rectHeight += rectHeightIncrease;
            xOffset += rectWidth;
        }
        // draw the right side rectangles, reset height of first
        rectHeight = size;
        for (int i = numRectangles / 2; i < numRectangles; i++) {
            if (i % 2 == 0) {
                g.setColor(Color.GRAY);
            } else {
                g.setColor(Color.WHITE);
            }
            g.fillRect(x + xOffset, y, rectWidth, length);
            rectHeight -= rectHeightIncrease;
            xOffset += rectWidth;
        }
    }
```

- set color to BLUE for background, 1 point
- draw background square correctly, 1 point
- determine number of rectangles correctly, 2 points
- determine width (fixed) of rectangles correctly, 2 points
- draw rectangles in loop
  - loop using number of rectangles, 3 points
  - Alternate color between GRAY and WHITE, 2 points
  - determine x coordinate of current rectangle, 2 points
  - determine height of current rectangle, 2 points
- correctly draw all rectangles, 5 points (can lose partial depending on severity of problem)

Other:

- incorrect calculation of y coordinate, -2 to -4 depending on severity
- hard coded numbers, -6
- not having correct height of left most rectangle on the right half. (In other words, when going left to right, the second rectangle that is the same size as the square. -3 to - 5 depending on severity