CS312 Fall 2017 Exam 2 Solution and Grading Criteria.

Grading acronyms:
AIOBE - Array Index out of Bounds Exception may occur
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
NPE - Null Pointer Exception may occur
OBOE - Off by one error. Calculation is off by one.
RTQ - Read the question. Violated restrictions or made incorrect assumption.

1. Code Trace:

A. `9 orns`
B. `ABBA BAA`
C. `false`
D `2 -1 true`
E. `-2, -1, 0 (no duplicates allowed)`
F. `No`
G. `17.0 18.5`
H. `TY`
I. `7`
J. `1213CS`
K. `e=_2.72ish  (underscore for single space)`
L. `AS BASIC`
M. `[1, 5, 6]`
N. `[2, 5, 10, 3]`

2. Program Logic (0.5 each)

	X <= y	y > z	x == 1
POINT A	**A**	**S**	**N**
POINT B	**S**	**N**	**S**
POINT C	**A**	**A**	**N**
POINT D	**S**	**S**	**S**
POINT E	**S**	**A**	**S**

3. Strings - 10 Points. Write a method that determines if two Strings start with the same N characters.

```
public static boolean sameStart(String s1, String s2, int n) {
    boolean same = s1.length() > n && s2.length() > n;
    int i = 0;
    while (i < n && same) {
        char c1 = s1.charAt(i);
        char c2 = s2.charAt(i);
        same = c1 == c2;
        i++;
    }
   return same;
}
```

return false / no work if one or both Strings don't have enough chars, 2 points
loop while still a match and chars to check. (for loop with return inside okay), 3 points
correctly access chars with charAt method, 1 point
correctly check chars at same spot equal, 2 points
return false as soon as mismatch found, 1 point
return true in cases where first n chars match, 1 point

using substring method or equals method on built up String: -7
using other disallowed methods: varies

4. Strings 10 Points. Write a method `reverseAndStretch`

Two obvious approaches, starting from back or starting at front and doing concatenation at correct spot

```
public static String reverseAndStretch(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        int times = i + 1;
        for (int j = 0; j < times; j++) {
            result = str.charAt(i) + result;
        }
    }
    return result;
}

public static String reverseAndStretchAlt(String str) {
    String result = "";
    for (int i = str.length() - 1; i >= 0; i--) {
        int times = i + 1;
        for (int j = 0; j < times; j++) {
            result = result + str.charAt(i);
        }
    }
    return result;
}
```

Create result as empty String: 1 point
outer loop for length of string: 2 points
inner loop for number of characters: 3 points
concatenate characters correctly, 3 points (must create correct resulting String)
return result, 1 point

5. Arrays 10 Points. Write a method `getDifferenceArray`.

```
public static int[] getDifferenceArray(int[] ar1, int[] ar2) {
    int resultLength = ar1.length;
    if (ar2.length < ar1.length) {
        resultLength = ar2.length;
    }
    int[] result = new int[resultLength];
    for (int i = 0; i < result.length; i++) {
        result[i] = ar1[i] - ar2[i];
    }
    return result;
}
```

find length of resulting array correctly: 2 points (Cannot use Math.min)
loop with correct bounds: 4 points (length() okay)
correctly set element at correct spot in resulting array to difference of elements from ar1 and ar2: 3 points
return result: 1 point

6. Programming. 16 points. Write a method that determine if a given digit occurs exactly a given number of times in an `int`. The method is named `digitPresent` and it returns a `boolean`.

```
public static boolean digitPresent(int num, int digit, int times) {
    // special case if num is initially 0
    if (num == 0) {
        return digit == 0 && times == 1;
    }
    // general case
    int digitCount = 0;
    while (num > 0 && digitCount <= times) {
        // get the current digit use remainder
        int currentDigit = num % 10;
        if (currentDigit == digit) {
            // Found one!
            digitCount++;
        }
        num /= 10;
    } // out of digits or digits > times, too many times!
    return digitCount == times;
}
```

handle special case in num is zero: 3 points
general case:

- variable to count number of occurrences: 1 point
- loop while num > 0: 3 points
- stop when we have more occurrences of digit than times: 2 points
- get curernt digit with %: 3 points
- compare current digit to target digit correctly: 1 point
- reduce num correctly: 2 points
- return correct result: 1 point

7. Scanners. 18 points. Write a complete method `linesWithWord`. The method accepts a `Scanner` already connected to a file and a target `String`.

```
public static void linesWithWord(Scanner sc, String tgt) {
    int lines = 0;
    int countOfTarget = 0;
    int max = 0;
    while (sc.hasNextLine()) {
        lines++;
        int timesTgtInLine = 0;
        Scanner lineScanner = new Scanner(sc.nextLine());
        while (lineScanner.hasNext()) {
            String token = lineScanner.next();
            if (token.equals(tgt)) {
                timesTgtInLine++;
            }
        }
        System.out.println(lines + ": " + timesTgtInLine);
        if (timesTgtInLine > max) {
            max = timesTgtInLine;
        }
        countOfTarget += timesTgtInLine;
    }
    System.out.println("total: " + countOfTarget);
    System.out.println("max times in line: " + max);
}
```

variables for line number: 1 point
variable for total number of times target appears: 1 point
variable for max initialized to 0 (or - value): 1 point
while loop for lines: 3 point
variable for number of times target appears in current line: 1 point
new Scanner for line: 1 point
loop for tokens in current line: 2 points
check if token is equal to target correctly: 1 point
print data for current line: 2 points
check if current times is new max: 2 points
update total count: 1 point
print total and max at end correctly: 2 points

common problems:
using arrays, not allowed - 3
padding with chars, - 4 (what if char you are padding with is the target?)
substring -2
equals method - 4
no accounting for positions of chars, -8