CS 312 – Exam 2 – Fall 2018

Your Name_____

Your UTEID _____

Circle your TA's Name: Aish Anthony Carla Daniel Dayanny

Fatima Kate Mallory Rebecca

Problem Number	Topic	Points Possible	Points Off
1	code trace	28	
2	program logic	8	
3	strings	10	
4	scanners & strings	17	
5	programming	12	
6	scanners	17	
7	arrays	8	
	TOTAL POINTS OFF:		
	SCORE OUT OF 100:		

Instructions:

1. You have 2 hours to complete the exam.
2. You must use a pencil to write your answers on the exam.
3. You may not use any electronic devices, notes, or other resources.
4. When code is required, write Java code. Limit yourself to the features from chapters 1 - 7 of the book and topics 1 - 23 in class.
5. Ensure you follow the restrictions of the question.
6. You may write and call your own helper methods.
7. The exam proctors will not answer questions. If you believe there is an error or a question is ambiguous, state your assumptions and answer based on those assumptions.
8. When you finish, show the proctor your UTID, turn in the exam and all scratch paper.

1. Evaluating Code. 28 points, 2 points each. Assume all necessary imports have been made.
If the snippet contains a syntax error or compiler error, answer **compile error**.
If the snippet results in a runtime error or exception answer **runtime error**.
If the code results in an infinite loop answer **infinite loop**.

A. What is output by the following code?

```
String a1 = "\tCS\n312\\312";
System.out.print(a1.indexOf('\\'));
```

Output A: _____

B. What is output by the following code?

```
String b1 = "Program";
String b2 = "pro";
String b3 = "gram";
String b4 = b2 + b3;
System.out.print(b1.toLowerCase() == b4);
```

Output B: _____

C. What is output by the following code?

```
String c1 = "Volleyball";
String c2 = c1.substring(5);
System.out.print(c2);
```

Output C: _____

D. What is output by the following code?

```
String d1 = "TeXaS";
d1.toUpperCase();
System.out.print(d1 + " " + d1.length());
```

Output D: _____

E. What is output by the following code?

```
String e1 = "track_and_field";
String e2 = e1.substring(2, 6);
System.out.print(e2);
```

Output E: _____

F. Given the following expression, how many of the 8 combinations of values for `p`, `q`, and `r` (all `boolean` variables) result in the expression evaluating to `false`?

```
(p || !q) && r
```

Answer F: _____

G. What is output by the following code? For this question only, use an underscore to represent spaces in the output. One underscore per space

```
double ge = 2.71828;
double g2 = 7.5;
System.out.printf("ge%6.2fg2%4.2f", ge, g2);
```

Output G: _____

H. What is output by the following code?

```
String h1 = "qq";
while (h1.length() < 25) {
      h1 = h1 + h1 + 'q';
}
System.out.print(h1.length());
```

Output H: _____

I. What is output by the following code?

```
String i1 = "wallace_color";
char i2 = i1.charAt(5 / 8 - 4);
int i3 = i1.indexOf("ae");
System.out.print(i2 + " " + i3);
```

Output I: _____

J. The following method does not compile. Why not?

```
public static void methodJ() {
      File f = new File("data");
      Scanner SCANNER = new Scanner(f);
      while (SCANNER.hasNextInt()) {
            System.out.print(SCANNER.next());
      }
}
```

Answer J: _____

K. What is output by the following code?

```
int[] dataK = {6, 3, 1, 7, -5, 8};
System.out.print(dataK[dataK[2]] + " " + dataK.length);
```

Output K: _____

L. What is output by the following code?

```
String sL = "vector";
methodL(sL, sL);
System.out.print(sL);

public static void methodL(String s, String t) {
      s = s + s.length();
      t = t + t;
      System.out.print(s.length() + t.length() + " ");
}
```

Output L: _____

M. What is output by the following code?

```
int[] m = new int[3];
methodM(m);
System.out.print(Arrays.toString(m));

public static void methodM(int[] m) {
      m = new int[m.length + 2];
      System.out.print(m[3] + " ");
}
```

Output M: _____

N. What is output by the following code?

```
int[] n = {-3, 4, -2, 6};
methodN(n);
System.out.print(Arrays.toString(n));

public static void methodN(int[] data) {
      data[2] = data.length;
      data[1]--;
      data[3] += data[0] + data[3] + data[2];
}
```

Output N: _____

2. Program Logic - 8 Points. Consider the following method. For each of the five points labeled by comments and each of the three assertions in the table, write whether the assertion is ***always*** true, ***sometimes*** true and sometimes false, or ***never*** true at that point in the code. Abbreviate ***always*** with an A, ***sometimes*** with an S and ***never*** with an N.

```
public static void mystery(int x) {
    if (x > 0) {
        int c = 0;
        int e = 0;
        // POINT A
        while (e < 5 && c < x) {
            // POINT B
            double a = Math.random();
            int y = ((int) (a * 10));
            c += y;
            if (y % 2 == 0) {
                // POINT C
                e++;
                // POINT D
            }
        }
        // POINT E
        System.out.println(c + " " + e);
    }
}
```

Abbreviate ***always*** with an A, ***sometimes*** with an S and ***never*** with an N.

	c < x	e < 5	e > 0
POINT A			
POINT B			
POINT C			
POINT D			
POINT E			

3. Strings 10 Points. Write a method `getEqualStart` that accepts two `String`s as parameters.

The method returns a `String` that is to the part of the two parameters that match exactly, character for character, at the **start** of the `String`s.

Examples of the parameters and the `String` returned by the method. The matching portion from the beginning of the two parameters are bolded to help visualize what the method does.

```
s1: ""                  s1: "REM"               s1: "R.E.M."
s2: "rem"               s2: "rem"               s2: "REM"
returns: ""             returns: ""             returns: "R"

s1: "rem sleep"         s1: "bands 2"           s1: "bands 2"
s2: "rem"               s2: "ands 2"            s2: "band"
returns: "rem"          returns: ""             returns: "band"

s1: "bands 2"           s1: "banding"           s1: "54321"
s2: "banding"           s2: "Banding"           s2: "section"
returns: "band"         returns: ""             returns: ""

s1: "Comp"
s2: "Comp"
returns: "Comp"
```

The only `String` methods you can use are the `charAt(int i)`, `length()`, `substring(int start)`, `substring(int start, int stop)`, and `String` concatenation.

Your method should not do unnecessary work.

Complete the method on the next page.

```
public static String getEqualStart(String s1, String s2) {
```

4. Scanners and Strings - 17 Points. Write a method that given a `Scanner` already connected to a file and a target `char`, returns the token in the file that contains the target `char` the largest number of times.

If none of the tokens in the file contain the target `char`, return an empty `String`.

If two or more tokens in the file are tied for the largest number of occurrences of the target `char`, return the token that appears nearest the beginning of the file.

The only methods you may use from the `Scanner` class are the `hasNext()` and `next()` methods.

The only methods you may use from the `String` class are the `length()` and `charAt(int index)` methods.

Do not use any other Java classes or methods.

Consider the following example file: (Source: https://en.wikipedia.org/wiki/ABBA) .

Tokens with an `'a'` are in bold to help illustrate the results of the method.

Tokens with an `'s'` are underlined to help illustrate the results of the method.

ABBA **are a** Swedish pop group formed in Stockholm in 1972 by **Agnetha Faltskog**, Bjorn **Ulvaeus**, Benny Andersson **and** Anni-Frid **Lyngstad**.

The group's **name** is **an acronym** of the first letters of their first **names**. They **became** one of the most **commercially** successful **acts** in the history of **popular** music, topping the **charts** worldwide from 1974 to 1982.

If the target `char` was `'a'`, the method would return the `String` `"are"`. There are many tokens that contain a single `'a'`, so the method returns the one that appears nearest the beginning of the file.

If the target `char` was `'s'`, the method would return the `String` `"successful"`. `"successful"` has three occurrences of `'s'`, more than any other token in the example.

If the target `char` was `'5'`, the method would return the `String` `""`. There are no tokens in the file that contain a `'5'`.

Complete the method on the next page.

```
/* sc is already connected to a file.*/
public static String tokenWithMost(Scanner sc, char tgt) {
```

5. Programming 12 Points. Write a method, `lines` that generates random line segments until the sum of the length of the generated line segments is greater than a given limit. The method returns the number of line segments that had to be generated before the limit was exceeded. The length of the generated line segments are always integers.

The method is passed 2 `int`s. The first represents the minimum possible length of the randomly generated line segments and second represents the maximum possible lengths of the randomly line segments. The parameters are named `min` and `max`.

`min` shall be less than `max`. min shall be greater than or equal to 0.

The method is passed a parameter `limit` which represents the total length the sum of the randomly generated line segments must exceed. `limit` shall be greater than 0.

Your method must create and use an object of type `Random` to generate random values.

You may use the `nextInt(int max)` method from the `Random` class.

Do not use any other Java classes or methods in your solution. **Not even methods from the `Math` class.**

The method header is `lines(int min, int max, int limit).`

The method **does not** produce any output; it simply returns the number of line segments that had to be generated before the sum of those line segments exceeded the given limit.

The following shows one example of lengths selected by the method call `lines(0, 5, 21).`

Again, your method shall not do any actual output.

```
segment length: 3 total: 3
segment length: 1 total: 4
segment length: 3 total: 7
segment length: 0 total: 7
segment length: 4 total: 11
segment length: 3 total: 14
segment length: 2 total: 16
segment length: 5 total: 21
segment length: 1 total: 22
```

Given the above series of random values the method would return 9.

Complete the method on the next page.

```
/* 0 <= min < max, limit >= 1 */
public static int lines(int min, int max, int limit) {
```

6. Scanners. 17 points. Write a method `aveOfDoubles` that given a `Scanner` already connected to a file, prints out the average of the doubles in each line in the file.

The method shall print out for each line, the line number, the sum of the tokens in that line that are doubles, and the average of the doubles in that line.

The method shall not count tokens that could be read as a double or an int. For example, the token `12` can be read as an int or a double, but your method shall not count such tokens as doubles.

You may create new `Scanner` objects connected to `String`s by calling the `Scanner` constructor. You may use the `hasNextLine, hasNextInt, hasNextDouble, hasNext, nextLine, nextInt, nextDouble,` and `next` methods from the `Scanner` class.

Do not use any other Java classes or methods.

Example file:

```
0.25    Mallory's section 51285 1.75  5.0

Kate   Dayanny     Daniel  12
37.5    Aish and Anthony
Larry 100 100 100
Fatima Carla 1.5  1.5  1.5 3.0  Rebecca
```

Expected output:

```
1: sum = 7.0, ave = 2.3333333333333335
2: no doubles
3: no doubles
4: sum = 37.5, ave = 37.5
5: no doubles
6: sum = 7.5, ave = 1.875
```

Complete the method on the next page.

```
/* sc is already connected to a file.*/
public static void aveOfDoubles(Scanner sc) {
```

7. Arrays. 8 points. Write a method named `equalsTarget` that given an array of `int`s and an `int` that represents a target sum, returns `true` if the sum of all the elements in the array equal the target sum, `false` otherwise.

Do not use any other Java classes or methods.

You can, of course, use the `length` field of the given array.

Examples and expected results:

array is `[]` and target is `0` -> returns `true`

array is `[]` and target is `12` -> returns `false`

array is `[]` and target is `-5` -> returns `false`

array is `[2, 5, 2, 5, 10]` and target is `24` -> returns `true`

array is `[2, 5, 2, 5, 10]` and target is `-5` -> returns `false`

array is `[0, 5, -5, 0, -5, 0 , 0]` and target is `0` -> returns `false`

array is `[0, 5, -5, 0, -5, 0 , 0]` and target is `-5` -> returns `true`

array is `[0, 0, 0, 0]` and target is `0` -> returns `true`

Complete the method, including the method header, on the next page.

```
// Write your method header and complete your equalsTarget method:
```