CS 312 – Exam 2 – Fall 2015

Your Name_____

Your UTEID _____

Circle your TA's Name: Aila CK Jialin Katherine B

KG Kris Megan Roman Sonika

Problem Number	Topic	Points Possible	Points Off
1	code trace	32	
2	scanners	15	
3	program logic	16	
4	strings	15	
5	arrays	12	
6	arrays and strings	15	
7	arrays	15	
	TOTAL POINTS OFF:		
	SCORE OUT OF 120:		

Instructions:

1. You have 2 hours to complete the test.
2. You must use a pencil on the exam. If you use a pen you lose 5 points.
3. You may not use a calculator or any other electronic device..
4. When code is required, write Java code. Ensure you follow the restrictions of the question. Limit yourself to the features from chapters 1 - 7 of the book and topics 1 - 25 in class.
5. You may break problems up into smaller methods. (In other words you can add helper methods.)
6. The 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.
7. When you finish, show the proctor your UTID, turn in the exam and all scratch paper.

1. Evaluating Code. 32 points, 2 points each. Assume all necessary imports have been made.
If the snippet contains a syntax error or compiler error, answer **syntax 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 = "gymnastics";
String a2 = a1.substring(3, 7);
System.out.print(a1 + " " + a2);
```

Output: _____

B. Are the two `boolean` expressions below logically equivalent? In other words given the same inputs do the two expressions always evaluate to the same `boolean` result? `b1, b2,` and `b3` are `boolean` variables.

Expression 1: `!(!b1 && (b2 || !b3))`
Expression 2: `b1 || (!b2 || b3)`

Answer: _____

C. What is output by the following code?

```
String c1 = "gates4*dell";
String c2 = c1.substring(4).toUpperCase().substring(1, 4);
System.out.print(c2 + " " + c2.indexOf("A"));
```

Output: _____

D. What is output by the following code?

```
int[] d = {5, 3, -1, 6, 4};
d[3] += d[1] + d[d.length -1];
d[1]--;
System.out.print(Arrays.toString(d));
```

Output: _____

E. What is output by the following code?

```
boolean[] e = new boolean[5];
for(int i = 1; i < e.length - 1; i++) {
     e[i] = !e[i - 1];
}
System.out.print(Arrays.toString(e));
```

Output: _____

F. What is output by the following code?

```
int[] f1 = {2, 0, 4, -3};
int[] f2 = new int[3];
f1[1] -= f2[1] - 4;
f2 = f1;
f2[3] *= 3;
f2[2] /= f1[1];
System.out.print(Arrays.toString(f1) + " " + Arrays.toString(f2));
```

Output: _____

G. What is output by the following code?

```
char[] g = {'a', 'A', 'X'};
g.length += 2;
g[3] = g[1];
g[4] = 'Z';
System.out.print(Arrays.toString(g));
```

Output: _____

H. What is output by the following code?

```
int[] h = {3, 1, 5};
methodH(h);
System.out.print(Arrays.toString(h));

public static void methodH(int[] d) {
      d[0] += d.length;
      d[1] *= d[2];
}
```

Output: _____

I. What is output by the following code?

```
int[] ii = {3, 1, 5};
methodI(ii);
System.out.print(Arrays.toString(ii));

public static void methodI(int[] ii) {
      ii[1] = ii[0] + ii[2];
      ii = new int[] {5, 2};
      ii[1] *= 3;
      System.out.print(" " + Arrays.toString(ii));
}
```

Output: _____

J. What is output by the following code?

```
int[] jj = {5, 2, 6, 3, 7};
jj[0] -= jj[1 + jj[1]];
jj[jj[3]] += jj[jj[2] - jj[0]] ;
System.out.print(Arrays.toString(jj));
```

Output: _____

K. What is output by the following code?

```
int[] k1 = {5, 4, 2, 1};
int[] k2 = {5, 3, 2, 0};
int[] k3 = k2;
k3[1]++;
k2[k1.length - 1] += 1;
System.out.print( (k1 == k2) + " " + (k3 == k2));
```

Output: _____

L. What is output by the following code?

```
String s2 = "DES20";
s2 = s2 + 12;
methodL(s2);
System.out.print(s2);

public static String methodL(String s2) {
      s2.substring(4);
      s2 = 1 + 2 + s2;
      System.out.print(s2 + " ");
      return s2;

}
```

Output: _____

M. List the possible values the following code will output.

```
Random rm = new Random();
int x = ((rm.nextInt(10) * 2) - 5) / 4;
System.out.print(x);
```

Possible Values: _____

N. What is output by the following code?

```
int[] n = {5, -2, 3, 4};
for(int i = 1; i < 5; i++) {
      n[i] = n[i - 1] - 3;
}
System.out.print(Arrays.toString(n));
```

Output: _____

O. What is output by the following code?

```
Color[] o = new Color[4];
o[1] = Color.ORANGE;
o[3] = Color.WHITE;
System.out.print(o[1].equals(o[3]) + " " + o[0].equals(o[2]));
```

Output: _____

P. What is output by the following code?

```
int[] p = {5, -1, 4, 2, 6};
for (int i = 1; i < p.length; i++) {
      if(p[i - 1] < p[i]) {
            p[i] *= 2;
      } else {
            p[i] -= p[i - 1] / 2;
      }
}
System.out.print(Arrays.toString(p));
```

Output: _____

2. Scanners. 15 points. Write a complete method `averageOfNonPositiveInts`. The method accepts a `Scanner` already connected to a file. The method returns the average of all the non-positive `int`s that appear in the file the `Scanner` is connected to. In other words all the `int`s less than or equal to 0. If there are no non-positive `int`s in the file then the method shall return 1.0.

For example, if the Scanner were connected to the following file:

```
line 1 has no non positive ints 12 54 123

12.5 -10 some more stuff -21 -37.62 - 0.1

-33 0 0 115 Any more ints???
last line of input -5 with 1 non positive int
```

the method would return -11.5. (-10 + -21 + -33 + 0 + 0 + -5) / 6 = -11.5

You may use the methods from the `Scanner` class. Do not use any other Java classes or methods. Do not use arrays.

```
// MORE ROOM FOR averageOfNonPositiveInts IF NEEDED
```

3. Program Logic 16 Points. Consider the following method. For each of the four points labeled by comments and each of the four assertions in the table, write whether the assertion is ***always*** true, ***sometimes*** true, 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 assertionPractice(int[] list) {
    if (list == null || list.length <= 1)
        return;

    int i = 0;
    int j = 0;
    int temp = list[i];
    int c = 0;
    // point A
    for (i = 1; i < list.length; i++) {
        temp = list[i];
        j = i;
        while (j > 0 && temp > list[j - 1]){
            // point B
            c++;
            list[j] = list[j - 1];
            list[j - 1] = temp;
            j--;
            // point C
        } // end of while loop
        // point D
    } // end of for loop
}
```

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

	c != 0	i == j	list[i] < list[j]	j == 0
POINT A				
POINT B				
POINT C				
POINT D				

4. Strings 15 Points. Write a method `getChoppedString` that builds a new, "chopped" version of a `String`. The method has three parameters: a `String` `str` and and two `int`s. The first `int`, `start`, specifies the position to start at in the original `String` and the second `int`, `skip`, specifies the number of characters to skip when building the resulting `String`.

You may assume the starting position is within bounds of the `String`.
In other words `0 <= start < str.length()`

You may assume `skip >= 2`.

Examples of `getChoppedString(String str, int start, int skip)`

```
getChoppedString("computer", 0, 3) -> returns "cpe"

getChoppedString("computer", 0, 4) -> returns "cu"

getChoppedString("computer", 1, 3) -> returns "our"

getChoppedString("computer", 1, 4) -> returns "ot"

getChoppedString("computer", 0, 8) -> returns "c"

getChoppedString("computer", 0, 9) -> returns "c"

getChoppedString("computer", 1, 8) -> returns "o"

getChoppedString("computer", 7, 2) -> returns "r"

getChoppedString("computer", 0, 2) -> returns "cmue"

getChoppedString("computer", 1, 2) -> returns "optr"

getChoppedString("computer", 2, 2) -> returns "mue"
```

You may use String concatenation and the String `charAt()` and `length()` methods.

You may not use any other Java classes or methods.

COMPLETE THE METHOD ON THE NEXT PAGE.

```
public static String getChoppedString(String str, int start, int skip)
```

5. Arrays 12 Points. Write a method `sumOfGaps`. The method has one parameter: an array of `int`s. The method returns the sum of the gaps between consecutive elements in the array. For this question we define the gap between two consecutive elements to be the second value minus the first value.

For example if we have the array `{6, 3, -2, 7, 15, 9}` the gaps are:

```
3 - 6    = -3
-2 - 3   = -5
7 - (-2) = 9
15 - 7   = 8
9 - 15   = -6
```

The sum of these gaps is `-3 + -5 + 9 + 8 + -6 = 3`

You may assume the array the method is passed has two or more elements.

Other examples:

```
sumOfGaps( {0, 3, 0, 3, 3, 1}) -> returns 1
sumOfGaps( {-1, -10}) -> returns -9
sumOfGaps( {15, -5, 20}) -> returns 5
sumOfGaps( {1, 6, 12, 18}) -> returns 17
```

You may not use any other Java classes or methods in your answer.

COMPLETE THE METHOD ON THE NEXT PAGE.

```
public static int sumOfGaps(int[] data) {
```

6. Arrays 15 Points. Write a method `numThatStartOrEndWithChar` that accepts two parameters, an array of `String` variables and a `char`. The method returns the number of `String`s in the array that start and / or end with the given `char`. Note, some elements of the array may store `null` and some of the `String`s may have a length of 0.

Examples. `numThatStartOrEndWithChar` is abbreviated as `ntsoewc` in these examples.

```
ntsoewc( {null, null, "", "AA", null}, 'n') -> returns 0

ntsoewc( {null, null, null, null}, 'n') -> returns 0

ntsoewc( { }, 'n') -> returns 0

ntsoewc( {null, "ABBA", "abba", "bbAAAb", "", ""}, 'A') -> returns 1

ntsoewc( {"ABBABB", "ABBA", "abba", "bbAAAbA"}, 'A') -> returns 3
```

You may use the `length` and `charAt` methods from the `String` class, but no other Java classes or methods.

COMPLETE THE METHOD ON THE NEXT PAGE.

```
public static int numThatStartOrEndWithChar(String[] vals, char c) {
```

7. Arays 15 Points. Write a method `removeTarget` that given an array of `int`s and a target `int`, creates and returns a new array that is the same as the given array **except** any element equal to the target `int` is not present. The length of the returned array equals the number of elements in the original array not equal to the target. The relative order of the elements not equal to the target `int` is the same.

Examples:

`removeTarget({2, 5, 2, 1, 6}, 3) returns {2, 5, 2, 1, 6}` Note, in this example the returned array is a new array that is a copy of the original array.

`removeTarget({2, 5, 2, 1, 6}, 2) returns {5, 1, 6}`

`removeTarget({}, 2) returns {}` Note, in this example the returned array is a new array that is a copy of the original array.

`removeTarget({2, 2, 2}, 2) returns {}`

`removeTarget({2, 5, 2, 1, 5, 5}, 5) returns {2, 2, 1}`

You may use native arrays but no other Java method or classes.
You may not use the static methods from the Arrays class.

COMPLETE THE METHOD ON THE NEXT PAGE.

```
public static int[] removeTarget(int[] data, int tgt) {
```