CS 312 – Exam 2 – Fall 2016

Your Name_____KEY_____

Your UTEID _____

Circle your discussion section time:

11am 12 noon 2pm

Problem Number	Topic	Points Possible	Points Off
1	code trace	30	
2	program logic	15	
3	scanners	16	
4	strings	15	
5	arrays	14	
6	arrays and strings	15	
7	strings	15	
	TOTAL POINTS OFF:		
	SCORE OUT OF 120:		

Instructions:

1. You have 2 hours to complete the test.
2. You may not use a calculator or any other electronic device.
3. 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.
4. You may break problems up into smaller methods. (In other words you can add helper methods.)
5. 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.
6. When you finish, show the proctor your UTID, turn in the exam and all scratch paper.

1. Evaluating Code (30 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 = "Chelsea";
String a2 = a1.substring(1, 4);
System.out.print(a2 + " " + a1.length());
```

Output: _____**hel 7**_____

B. What is output by the following code?

```
String b1 = "longhorns";
String b2 = a1.substring(4);
System.out.print(b2 + " " + b1.charAt(2));
```

Output: _____**horns n**_____

C. What is output by the following code?

```
String c1 = "cs";
String c2 = "ut";
String c3 = "utcs";
String c4 = c2 + c1;
System.out.print(c3 == c4);
```

Output: _____**false**_____

D. 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? `x, y,` and `z` are `int` variables.

Expression 1: `(x < y) || (y == z)`
Expression 2: `!((x < y) && (y != z))`

Answer: _____**no**_____

E. Given the following expression, how many of the 8 combinations of values for `e1`, `e2`, and `e3` (all `boolean` variables) cause the expression to evaluate to `false`?

```
(e1 || !e2) && e3
```

Answer: _____**5**_____

F. What is output by the following code?

```
String f1 = "BDM8";
f1.toLowerCase();
f1 += "**";
System.out.print(f1);
```

Output: _____**BDM8****_____

G. What is output by the following code assuming the user types in the following:

12 ONE

```
Scanner sc = new Scanner(System.in);
int x = sc.nextInt();
int y = sc.nextInt();
System.out.print(x + " " + y);
```

Output: _____**runtime error**_____

H. What is output by the following code? *For this question only* use an underscore character, _, to indicate any spaces in the output. One underscore per space.

```
double h1 = 12.37;
System.out.printf("val = %6.3f", h1);
```

Output: _____**val_=_12.370**_____

I. What is output by the following code?

```
int[] i1 = new int[5];
System.out.print(i1.length + " " + i1[3]);
```

Output: _____**5 0**_____

J. What is output by the following code?

```
int[] data1 = {5, -3, 7, 2, 4};
int j1 = 3;
data1[j1] += data1[j1] + j1;
data1[data1.length - 1] = data1[3] - data1[4] + data1[1];
System.out.println(Arrays.toString(data1));
```

Output: _____**[5, -3, 7, 7, 0]**_____

K. What is output by the following code?

```
int[] k1 = {5, 3, 2, -1};
int x2 = 3;
int y2 = x2 * 2;
if (k1[y2] < 5 && y2 < k1.length) {
      k1[1] = 12;
}
System.out.print(Arrays.toString(k1));
```

Output: _____**runtime error**_____

L. What is output by the following code?

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

public static void methodL(int[] data) {
      data[1] += 2;
      data[3] = data[1] + data[2];
}
```

Output for L: _____**[5, 7, 3, 13]**_____

For parts M, N, and O consider the following method:

```
public static int[] create(int val) {
      int[] result = new int[val + 2];
      for (int i = 0; i < result.length; i++) {
            result[i] = val * i + i * i;
      }
      return result;
}
```

M. What is output by the following code?

```
int[] m = create(-2);
System.out.print(Arrays.toString(m));
```

Output: _____**[]**_____

N. What is output by the following code?

```
int[] n = create(1);
System.out.print(Arrays.toString(n));
```

Output: _____**[0, 2, 6]**_____

O. What is output by the following code?

```
System.out.print(Arrays.toString(create(3)));
```

Output: _____**[0, 4, 10, 18, 28]**_____

2. Program Logic (15 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() {
      Random r = new Random();
      int y = 10;
      int z = 0;
      // POINT A
      while (y != 0) {
            // POINT B
            y = r.nextInt(23);
            if (y % 4 == 0) {
                  z++;
                  // POINT C
                  y--;
            }
            // POINT D
            y--;
      }
      // POINT E
      System.out.println("z = " + z);
}
```

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

	z == 0	y == 0	y % 4 == 0
POINT A	**A**	**N**	**N**
POINT B	**S**	**N**	**S** (initially y = 10: false, if y picked to be 5, decrement to 4, 4 != 0 at top of while loop, 4 % 4 == 0: true)
POINT C	**N**	**S**	**A**
POINT D	**S**	**N**	**N**
POINT E	**S** (If y is picked to be 0 in the loop it is decremented to -1 in the if statement.)	**A**	**A**

3. Scanners (16 points): Write a complete method `ratioOfIntsToDoubles`. The method accepts a `Scanner` already connected to a file. The method returns the ratio of the number of integers in the file to the number of doubles in the file.

If there are no doubles in the file the method shall return `-1.0`.

A token that could be read as an `int` or a `double` shall be read as an `int`. For example the token `12` could be read as an `int` or a `double`, but your method shall read it as an `int`.

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

```
Data data data 12 5      I need data

One cannot make bricks (  0.3  ) without clay! 12
You are 0 (<- that's a zero, not an oh.) help.
17        -2.5      -12
Last line of data with two numbers  0.5    37.14
```

The file in the example has 6 integers (`12  5  12  0  17  -12`)
and 4 doubles (`0.3  -2.5  0.5  37.14`).

Given that file the method would return 1.5. 6 / 4 = 1.5

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

Do not use arrays.

Complete the method on the next page.

```
public static double ratioOfIntsToDoubles(Scanner sc) {
  int nDouble = 0;
  int nInt = 0;
  while (sc.hasNext()) {
    // Check for ints first (since a double can be an int)
    if (sc.hasNextInt()) {
      nInt++;
    } else if (sc.hasNextDouble()) {
      nDouble++;
    }

    // At this point, consume the token no matter what
    // (we don't care what it is, just want to go to the next
    // thing).
    sc.next();
  }

  // Make sure there are more than 0
  if (nDouble + nInt == 0) {
    return -1;
  }

  // Return the ratio
  return (double)nInt / nDouble;
}
```

4. Strings (15 points): Create a method `getStretchedString` that accepts two parameters, a `String` and an `int`. The method creates and returns a new `String` with each character repeated the given number of times.

Examples of calls to `stretchString`:

```
getStretchedString("abc", 3) returns "aaabbbccc"

getStretchedString("", 3) returns ""

getStretchedString("Example 3", 2) returns "EExxaammppllee  33"

getStretchedString("aaDa", 4) returns "aaaaaaaaDDDDaaaa"
```

Assume the int parameter `num` is greater than 1. `(num > 1)`

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

Do not use any other Java methods or classes.

Complete the method on the next page.

```
public static String getStretchedString (String str, int num) {
  String toRet = "";
  for (int i = 0; i < str.length(); i++) {
    for (int j = 0; j < num; j++) {
      toRet += str.charAt(i);
    }
  }

  return toRet;
}
```

5. Arrays (14 points): Write a method `numLessThanPrevious`. The method has one parameter: an array of `int`s. The method returns the number of elements in the array that are less than the element just preceding it in the array.

Examples of results given various arrays:

```
[] returns 0

[1]  returns 0

[2, 1] returns 1 ( 1 < 2 )

[1, 5, 10, 12, 37] returns 0

[1, 1, 1, 1, 1] returns 0

[5, 2, 0, -12, -101, 10, 5] returns 5
                                (2 < 5, 0 < 2, -12 < 0, -101, -12, 5 < 10)
```

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

Do not create any new arrays.

The parameter is unaltered by this method.

Complete the method on the next page.

```
public static int numLessThanPrevious (int[] data) {
  int numLess = 0;

  // Start with i at 1 so we can compare to thing previous.
  for (int i = 1; i < data.length; i++) {
    if (data[i] < data[i - 1]) {
      numLess++;
    }
  }
  return numLess;
}
```

6. Arrays (15 points): Write a method `noDuplicates` that accepts one parameters, an array of `String` variables. The method returns `true` if each `String` in the array appears exactly once in the array. In other words the method returns `true` if there are no duplicate `String`s in the array.

Examples of results given various arrays:

```
[]   returns true

["the"]   returns true

["the", "The"] returns true

["a", "aa", "b", "the", "", "REM"] returns true

["", "FYC", "", "the", "THE"] returns false, "" repeated

["the", "the", "the", "the", "the"] returns false

[["a", "aa", "b", "the", "", "REM", "aa", "ELO"]  returns false,
                                                        "aa" repeated
```

You may assume the array sent to the method DOES NOT contain any elements equal to `null`.

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

Do not alter the original array.

Do not create any new arrays.

You method should not do any unnecessary work. In other words it should be as efficient as possible given the constraints of the question.

Complete the method on the next page.

```
public static boolean noDuplicates(String[] data) {
  for (int i = 0; i < data.length; i++) {
    for (int j = i + 1; j < data.length; j++) {
      // If this is ever true, return early.
      if (data[i].equals(data[j])) {
        return false;
      }
    }
  }

  // If I've gotten to this point, there are no duplicates.
  return true;
}
```

7. Strings (15 points): Write a method `matchingChars`. The method accepts two `String`s and a `char` as parameters. The method returns true if the two Strings contain the given character in the same place **for all occurrences of the given character**.

Examples of `matchingChars(String s1, String s2, char ch)`

```
s1   "dad day"      returns true
s2   "cab sad"
ch   'a'

s1   "dad day"      returns false, no match for second 'a' in s1
s2   "cab"
ch   'a'

s1   "dad"          returns false, no match for second 'a' in s2
s2   "cab also"
ch   'a'

s1   "dad day"      returns true, there are no *'s so all occurrences match
s2   "cab sad"
ch   '*'

s1   "that old band ABBA"          returns false, case sensitive
s2   "that new band abba??"
ch   'a'

s1   "example**with some***non*letters*"          returns true
s2   "but has**spacesins***ome*places!*!!  !!"
ch   '*'
```

You may use 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 boolean matchingChars(String s1, String s2, char ch) {
  // If both Strings succeed, we'll return true.
  return matchingSingle(s1, s2, ch) && matchingSingle(s2, s1, ch);
}

/**
 * This function only checks against the length of a single String.
 * We'll need to call it for the other String later, but this simplifies
 * things extremely.
 */
public static boolean matchingSingle(String s1, String s2, char ch) {
  for (int i = 0; i < s1.length(); i++) {
    if (s1.charAt(i) == ch) {
      // If we've found a character, make sure it's the same
      // in the other String.
      if (!isMatching(s1, s2, i)) {
        return false;
      }
    }
  }

  // Good so far for this String.
  return true;
}

public static boolean isMatching(String s1, String s2, int i) {
  // Make sure this doesn't go over the end of the String.
  if (i >= s1.length() || i >= s2.length()) {
    return false;
  }

  // boolean zen.
  return s1.charAt(i) == s2.charAt(i);
}
```