CS 312 – Midterm 2 – Spring 2013

Suggested Solution and Grading Criteria

Problem Number	Topic	Points Possible	Points Off
1	array trace	12	
2	conditionals	9	
3	strings	10	
4	program logic	9	
5	programming	12	
6	scanner	8	
TOTAL POINTS OFF:			
SCORE OUT OF 60:			

Grading Acronyms:

NAP = No Answer Provided

LE = Logic Error

GCE = Gross Conceptual Error, based on answer did not understand question.

OBOE = Off By One Error

BOD = Benefit Of the Doubt

GACK = poor style or approach, but no points off

NN = Not Necessary. Generally no points off.

1. Arrays. 12 points. For each code snippet what is output when the code is run? If the snippet contains a syntax error answer "syntax error". If the snippet results in a runtime error or exception answer "runtime error".

Correct answer or -2. No partial credit

A.
```
int x = 3;
int y = 2;
int[] data1 = new int[x * y + 2];
System.out.print(data1.length);
```

OUTPUT: **8**

B.
```
double[] data2 = {0.1, 3.5, 1.5, 6.1, 0.5};
System.out.print(data2[1] + " " + data2[data2.length/2]);
```

OUTPUT: **3.5 1.5**

C.
```
int[] data3 = new int[6];
System.out.print(data3[0] + " " + data3[3]);
```

OUTPUT: **0 0**

D.
```
String[] names = new String[5];
System.out.print(names[0].length() + " " + names[2].length());
```

OUTPUT: **runtime error (due to null pointer exception)**

E.
```
int[] data4 = {5, 1, 6, 2, 3};
data4[3] += 3;
data4[2] = data4[3] * data4[1];
data4[1] = 7;
System.out.print(data4[1] + " " + data4[2] + " " + data4[3]);
```

OUTPUT: **7 5 5**

E.
```
int[] data5 = new int[12];
System.out.print(data5[data5.length - 2 * data5.length]);
```

OUTPUT: **runtime error (due to array index out of bounds exception)**

2. Conditionals. 9 points. If you follow the news, Cyprus, the country, is working on a plan to bailout its banks. One part of the current plan involves "taxing" (stealing?) money from citizens and corporations deposited in the bank. (Imagine if your bank announced it was simply going to tax (take) 10% of the money in your account. You go to sleep with $1000 dollars in your account and wake up the next morning, without having spent a dime yourself, and your balance is $900.)

In one version of the plan, deposits are taxed based on the amount in the account with different percentages.

Balances of greater than 0 and <= 100,000 Euros: 5% tax
Balances greater than 100,000 Euros and <= 500,000 Euros: 5,000 plus 10% tax on the amount over 100,000
Balances greater than 500,000 Euros: 45,000 plus 15% tax on the amount over 500,000

Complete a method that takes in a single parameter, an account balance in Euros expressed as a `double` and returns the tax on the account, expressed as a `double`. If the parameter is less than zero return `-1.0`.

Examples:
```
given a parameter equal to 50000.0 the method returns 2500.0
given 200000.0 the method returns 15000.0
given 1000000.0 the method returns 120000.0
```

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

```
public static double calcTax(double balance) {
    double result = 0;
    if(balance <= 0)
        result = -1;
    else if(balance <= 100000)
        result = .05 * balance;
    else if(balance <= 500000)
        result = 5000 + (balance - 100000) * .1;
    else
        result = 45000 + (balance - 500000) * .15;
    return result;
}
```

header correct with parameter: 1
handle case balance < 0: 1
handle balance between 0 and 100,000: 2
handle balance between 100,000 and 500,000: 2
handle balance greater than 500,000: 2
return correct answer: 1

Biggest problem: not subtracting correct amount from balance for two highest tax rates. Failure to do this results in a tax that is too large.

3. Strings. 10 points. Write a method named `newString` that accepts two parameters, a `String` and a `char`. The method returns a `String` with all instances of the char parameter removed and all instances of the asterisk character, `'*'`, removed. The order of other characters is the same.

Examples:

```
newString("C*S*42*9", '4') returns "CS29"
newString("****3333", '3') returns ""
newString("CS312", 'a') returns "CS312"
```

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

```
public static String newString(String src, char tgt) {
    String result = "";
    for(int i = 0; i < src.length(); i++) {
        char ch = src.charAt(i);
        if(ch != tgt && ch != '*')
            result += ch;
    }
    return result;
}
```

loop with correct bounds : 2
access char correctly: 1
logic correct so tgt and '*' not added: 2
build resulting string correctly: 4
return correct string: 1

common problems:

wrong condition ch != tgt || ch != '*'
think about it. if tgt is NOT a '*' that boolean expression is ALWAYS true

printing stuff: description said nothing about printing

using a Scanner. ????/

using methods besides charAt and length

not using String concatenation correctly str.charAt(4) = ""; does not work

4. Program Logic (9 points) Consider the following method. For each of the three points labeled by comments and each of the three 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. Assume the elements in data are sorted in ascending order and that the array has at least one element.

```
// elements in data must be in ascending order. data.length > 0
public static int assertionPractice(int[] data, int tgt){
     int s = 0;
     int h = data.length - 1;
     int r = -1;
     int m = 0;
     while(r == -1 && s <= h) {
          m = (s + h) / 2;
          if(data[m] == tgt) {
               // POINT A
               r = m;
          }
          else if(data[m] < tgt)
               s = m + 1;
          else
               h = m - 1;
          // POINT B

     }
     // POINT C
     return r;
}
```

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

	r == -1	s > h	data[m] != tgt
Point A	A	N	N
Point B	S	S	S
Point C	S	S	S

5. Programming. 12 points. Write a method that simulates rolling two six-sided dice. The method accepts two parameters, a target number between 2 and 12, and a required goal, an `int` greater than 0. The method simulates rolling a pair of six sided dice until the number of times the target number has been rolled equals the goal.

For example if the target number is 7 and the goal is 3, the method "rolls" the dice until 7 has been rolled three times. The method returns the total number of times the dice were rolled to achieve the goal.

Consider this example with a target of 7 and the required number of 7's equal to 3. If the random dice rolls are

`12, 5, 2, 8, 5, 7, 6, 7, 5, 5, 7`

The method would return 11 because it took 11 rolls of the dice to get three 7's.

Recall how to create an object of type `Random`:

```
Random r = new Random();
```

and the method from the `Random` class, `nextInt(n)` that returns and `int` from 0 to n - 1.

```
public static int timesToRoll(int tgt, int requiredTimes) {
    Random r = new Random();
    int timesTgtRolled = 0;
    int totalRolls = 0;
    while(timesTgtRolled != requiredTimes) {
        totalRolls++;
        int roll = r.nextInt(6) + r.nextInt(6) + 2;
        if(roll == tgt)
            timesTgtRolled++;
        // System.out.print(roll + ", "); for debugging
    }
    return totalRolls;
}
```

correct method header: 1
variables to track total rolls and number of times tgt rolled: 1
create and use Random object correctly: 1
while loop with correct condition: 2
simulate rolling 2 six sided dice correctly: 3
check roll equals tgt correctly: 2
update times target rolled correctly: 1
return correct value: 1

common problems:

incorrectly creating Random object
int roll = r.nextInt(11) + 2; -> not the same as 2 six sided dice.
printing values
returning goal instead of total number of rolls

6. Scanner. 8 Points. Write a method that accepts a `Scanner` object as a parameter. The `Scanner` is already connected to a file. The file contains **at least** one token that is an integer. There may be other, non-integer tokens in the file. The method shall return the average of the integers in the file.

For example, if the `Scanner` object was connected to the following file,

```
10    cat    20
30
50   20cat20
cat
dog  32
```

then the method would return `28.4`. (142 / 5).

Note your method does nothandle `int`s that may be "embedded" in other tokens. For example, do not count the 20's in a `String` such as `"20cat20"`.

You may only use methods from the `Scanner` class. Do not use any other classes or methods in your solution.

```
public static double averageFile(Scanner sc) {
    double total = 0;
    int count = 0;
    while(sc.hasNext()) {
        if(sc.hasNextInt()) {
            total += sc.nextInt();
            count++;
        }
        else {
            sc.next();
        }
    }
    return total / count;
}
```

number of ints and total of ints variables: 1
while with hasNext: 2
check next int correctly: 1
update variables correctly: 1
skip tokens that are not ints correctly: 2
return correct value: 1

Common problems:

A LOT of people wrote

while(sc.hasNextInt())

this stops as soon as the first non int token is found. All of the ints that may follow that are skipped! -4 for this mistake.

Bonus Question 1: From the class on the Friday before spring break. Two UTCS faculty members have won the Turing Award, the "Nobel Prize" of computer science. What are their last names?

EMERSON DIJKSTRA