COP 3330 Suggested Exercises for Week 4

Static Methods

1) Write a static method that takes in a String str and a character c, and returns the number of times c appears in s. For example, count("hello", "l") should return 2. Write the method and call it with several test cases you make on your own. Here's the method prototype:

```
public static int count(String str, char c);
```

2) Write a static method that takes in a positive integer n and returns the sum of the first n positive perfect squares. (Note: the first few positive perfect squares are 1, 4, 9, 16,...) To get some practice with loops, please use a loop in your method instead of the formula for this sum. Test your method by calling it with several test cases you make up. Here's the method prototype:

```
public static int sumSquares(int n);
```

3) Edit the posted program PrintPatterns to add a menu of choices with patterns to print out where the user is presented the menu and then chooses which design to print. Once they choose the design, the user is asked appropriate questions to specify the exact design to be printed. After your program prints the requested design, it should reprompt the user with the menu. This should continue until the user chooses to quit.

4) A nice game for students to practice their arithmetic is a game where students get practice questions that are randomly generated (either addition, subtraction or multiplication). Design such a program utilizing static methods. Feel free to add any bells and whistles you'd like to!

5) One algorithm to find an approximation to the square root of a positive real number x is to run a binary search of the answer for 1 fixed number of iterations. The way this works is by setting initial low and high bounds for the search. If $x > 1$, then the low bound is 1 and the high bound is x. If $x < 1$, then the low bound is x and the high bound is 1. For each iteration of the search, guess that the square root is halfway in between low and high. Then see if the guess is too low or too high. For example, if $x = 4.0$ initially, the low and high start off as 1 and 4, respectively. Our first guess is $(1+4)/2.0 = 2.5$. Notice that $2.5^2 > 4$. This means that the highest possible value for the square root is 2.5. (Whatever the square root is, it's less than 2.5.) If you repeat this process 100 times, each time either changing the low bound or the high bound to the previous guess, then the low and high bounds will be extremely close to each other and either can be used as the approximation to the answer. The method prototype is given below. Write the method and then call it with various test cases to see how well it works:

```
public static double mysqrt(double x);
```

Arrays

6) Write a program that reads in test scores from standard input (first integer entered is number of scores, each of the following integers entered are the scores, one per line). and then prints out a histogram only printing out the rows corresponding to scores that are in the data. For example, if the tests scores in the file were 55, 80, 80, 95, 95, 95 and 98, the output to the screen should look like:

```
55*
80**
95***
98*
```

7) Write a static method that takes in a reference to an integer array and randomly shuffles the contents of the array. One way to do this is to choose two random indexes into the array and swap the contents 10n times, where n represents the length of the array. Test your method and make sure it passes the "eye test." Here's the method prototype:

```
public static void shuffle(int[] arr);
```

8) Write a static method that takes in a reference to an integer array, arr, and returns the number of inversions in the array. An inversion in an array, arr, is an ordered pair (i, j) such that i < j but arr[i] > arr[j]. Test your method on several arrays.

```
public static int numInversions(int[] arr);
```

9) Write two methods to calculate the average and standard deviation of an array of values. The method prototypes are as follows:

```
public static double average(double[] values);
public static double stddev(double[] values);
```

Note that the second method should call the first one. Please look up the definition of standard deviation if you need to and test your methods on test cases you generate.

10) Write a program that reads in information about a set of pizza orders from standard input and calculates the total price of each order. In particular, the input file will be formatted as follows:

1. The first line of the input file will contain a single positive integer, *n (n < 100)*, specifying the number of different pizza orders.

2. The next *n* lines will each contain one positive real number specifying the price for that corresponding pizza. (The first line will have the price for pizza 0, the next line for pizza 1, etc.)

3. The following line will contain a single positive integer *k* ($k < 50$) representing the number of orders you have to evaluate.

4. The next *k* lines will contain information about each of the *k* orders with one order per line.

5. Each of these lines will contain *n* non-negative integers, representing how many of those pizzas, respectively are in the order. (For example, if n=5 and the line contains the values 0 0 6 0 9, then the order contains 6 slices of pizza #2 and 9 slices of pizza #4.)

For each of the *k* test cases, output a line with the following format:

On day X, you will spend $YY.YY on pizza.

where X is the test case number (starting with 1), and YY.YY is the price of the pizza displayed with exactly 2 digits after the decimal. (Note: The price may exceed 100 dollars, so YY simply represents a dollar amount and not the exact number of digits in that dollar amount.)

Sample Input File (pizza.txt)

```
5
3.00
3.50
4.50
5.00
6.00
3
1 1 1 1 1
0 0 2 1 6
1 0 3 2 3
```

Sample Output (Corresponding to Sample Input File)

```
On day 1, you will spend $22.00 on pizza.
On day 2, you will spend $50.00 on pizza.
On day 3, you will spend $44.50 on pizza.
```