SI@UCF: Java Class
Homework Assignments: Pizza, Palindromes, Stocks

Reading Input From a File

To read input from a file instead of from the keyboard, just set up your Scanner as follows:

```
Scanner stdin = new Scanner(new File("file.txt"));
```

Replace file.txt with the name of the actual file you want. The file must be in the same directory as the .class file that gets created once your program is compiled.

In addition, since you are using file, you'll need a new import statement:

```
import java.io.*;
```

Due to this import, the declaration of main changes a bit:

```
public static void main(String[] args) throws Exception {
```

Finally, at the end of main close the file:

```
stdin.close();
```

Problem A: Group Dinner at Lazy Moon

You have finally gotten sick of always ordering the same exact pizza at Lazy Moon. You are now looking to diversify your orders. Also, you and your friends have determined it's faster for one person to order for everyone than for everyone to order separately. Thus, each time you go, either you or one of your friends will put in several orders.

To simplify this problem (as many restaurants do), each different pizza order will be given a number (starting from 0) and each of these orders will have a price. Your program will read in this information from a file named "pizza.txt" and then use it to determine the prices of different orders of pizza. The information for these orders will ALSO be in the file. The specifications for the file format are given below.

Input Specification

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

Output Specification

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

```
On day X, you will spend $YY.YY at Lazy Moon.
```

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 at Lazy Moon.
On day 2, you will spend $50.00 at Lazy Moon.
On day 3, you will spend $44.50 at Lazy Moon.
```

Problem B: Palindromes

A palindrome is a word that reads the same forwards as backwards. For this problem, a palindrome will be defined as a sentence that has words that are the same whether the sentence is read forwards or backwards. (Using this definition, the sentence, "I am I" is a palindrome.)

The input file for this program, palindrome.txt, will provide several test cases. For each of these test cases you simply have to determine whether or not the sentence provided is a palindrome or not.

Input Specification

1. The first line of the input file will contain a single positive integer, *n (n < 100)*, specifying the number of sentences you have to test.
2. The next *n* lines will each contain one positive integer, *w (w < 100)* specifying the number of words for that sentence. Each of the words in the sentence follows in order, separated by spaces. Each word will be comprised only of lowercase letters.

Output Specification

For each test case, output a line with the following format:

```
Test case k: ANS
```

where k is replaced with the number of the test case, starting with 1, and ANS is replaced with either "YES" or "NO".

Sample Input File (palindrome.txt)

```
4
3 i am i
8 this sentence makes no no makes this sentence
8 this sentence makes no no makes sentence this
4 madam i am adam
```

Sample Output (Corresponding to Sample Input File)

```
Test case 1: YES
Test case 2: NO
Test case 3: YES
Test case 4: NO
```

Problem C: Stocks

Stock companies like to make claims of the following form:

This year, we have seen our stock gain at least X points in a single day Y times.

You will write a program that helps a company make these claims. Your program will read in *n* days worth of stock values for the company and then be asked several questions of the form above based on those *n* prices from a file called stock.txt. Basically, it will be given a number, (such as .25), and be asked to determine the number of times in the run of *n* days, the stock rose by that value or more on a single day.

Input Specification

1. The first line of the input file will contain a single positive integer, *n (n < 1000)*, specifying the number of days for which we have stock data.
2. The next *n* lines will each contain one positive real number, specifying the price of the stock on that day.
3. This will be followed by a line with a single positive integer *q*, representing the number of questions that will be posed.
4. The next *q* lines will each contain one positive real number, specifying a price increase. (This is the value for which you have to determine how many days showed a price increase of at least this value.)

Output Specification

For each test case, output a line with the following format:

```
Our stock gained at least X points in a single day Y times.
```

where X is the real number specified in the file for that test case, and Y is the number of days for which the stock increased by at least that value.

Sample Input File (stock.txt)

```
10
39.75
40.00
41.25
40.00
39.85
40.00
40.15
40.25
40.40
42.00
3
.5
1.0
.01
```

Sample Output (Corresponding to Sample Input File)

```
Our stock gained at least .5 points in a single day 2 times.
Our stock gained at least 1.0 points in a single day 2 times.
Our stock gained at least .01 points in a single day 7 times.
```