5.2 Practice Programs

Sample solutions to these problems will be included by the beginning of 2013 at the following website:

http://www.cs.ucf.edu/~dmarino/ucf/transparency/cop3223/book/

1) Test Average Problem: Write a program that reads in a set of test scores from a file called "test.txt" and prints out to the screen the average of those test scores. The first line of the file will contain a positive integer *n*, representing the number of test scores that follow in the file. The following *n* lines will contain one test score each. Each test score is guaranteed to be an integer in between 0 and 100, inclusive.

2) Test Average Problem Revision #1: Write a program that reads in the same file as above, but also calculates the minimum and maximum test scores, as well as which test number both of those were. (For example, if the test scores in the file were 70, 85, 99, 67 and 83, then the minimum test score was 67 on test #4 and the maximum test score was 99 on test #3.) Have your program print out this information to the screen.

3) Test Average Problem Revision #2: Write a program that reads in test information for several students from a file and outputs each student's test average. (Each of the 10 tests is worth 10% of the grade.) Read the input from the file "testscores.txt". The file format is as follows:

The first line of the file contains a single positive integer, *n (< 100)*, representing the number of students in the class. The next *n* lines will contain 10 integers each separated by spaces, representing that student's ten test scores. Each of these 10 integers will in between 0 and 100, inclusive.

Write out one line of output to the screen for each student with the following format:

```
Student k Test Average: X
```

where k is the number of the student, starting with one and X is a double representing the average of their test grades, printed to one decimal place.

Sample Input File

```
3
100 100 100 100 100 100 100 100 100 100
90 100 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 99
```

Corresponding Output

```
Student 1 Test Average: 100.0
Student 2 Test Average: 91.0
Student 3 Test Average: 90.9
```

4) Test Average Problem Revision #3: Consider a new input file format for question #3, where the first line of the file contains two positive integers, *n*, and *k*, separated by integers, where *k* represents the number of tests each student has taken. The following *n* lines contain *k* integers in betweem 0 and 100, inclusive, representing the *k* test grades for each student. Use the same output format as the first question, but print out the averages to two decimal places.

Sample Input File

```
3 4
100 100 100 100
92 97 93 93
88 100 92 93
```

Corresponding Output

```
Student 1 Test Average: 100.0
Student 2 Test Average: 93.75
Student 3 Test Average: 93.25
```

5) Test Average Problem Revision #4: Edit the program written for number two to write the output to the file, "testscores.out", instead of the screen.

6) A file *nintendo.in* contains a single non-negative integer on each line representing donations (in dollars) to your Nintendo game buying fund. The end of the list will be signified by the value 0 on the last line of the file. (All other integers in the file will be positive.) A Nintendo game costs $50 to buy. Read in the file of donations and output a statement listing the maximum number of games you can buy as well as how much money you will have left over after buying that many games.

7) Read in any text file and output the total number of uppercase letter and lowercase letters in that textfile. Also include a count for how many non-alphabetic characters there are. Remember that you can read in a file one character at a time and that you can compare characters using the regular relational operators (>= , <=, etc.).

8) In this problem, you'll decide if three sides make a triangle or not. The input will come from the file, "triangle.txt". The first line of the input file will contain a single integer, *n*, representing the number of potential triangles to test. The following *n* lines will contain three positive integers (less than 1000000), separated by spaces, representing lengths in inches. For each potential triangle determine if the three values could be the lengths of sides of a triangle or not. For each case, format the output in one of the two following ways:

```
Case #k: Is a triangle!
Case #k: Not a triangle.
```

where k is the potential triangle in question, starting at 1.

Sample Input File

```
2
3 4 5
8 2 6
```

Corresponding Output

```
Case #1: Is a triangle!
Case #2: Not a triangle.
```

9) Serena likes all of her packages of cookies to have the same number of cookies in them. For this problem, you'll read in several test cases of her cookie packages and have to determine if she likes them or not. Read the input from the file "cookies.txt". The format of the file is as follows:

The first line of the file will contain a positive integer, *n*, representing the number of test cases to process. The following *n* lines will have one test case each. Each of these lines will start with a single positive integer, *c*, representing the number of packages of cookies for that test case. This is followed by *c* integers representing the number of cookies in each package.

For each test case, print out a single line with one of the two following formats:

```
Case #k: Serena likes her boxes of cookies!
Case #k: Serena doesn't like her boxes of cookies.
```

where k is the test case, starting with 1.

Sample Input File

```
2
3 4 5 4
8 2 2 2 2 2 2 2 2
```

Corresponding Output

```
Case #1: Serena doesn't like her boxes of cookies.
Case #2: Serena likes her boxes of cookies!
```

10) Write a program that calculates the maximum volatility of a stock. The input will come from a file. For each stock, you'll be given several days of data. Your job will be to calculate the greatest change in the stock between two consecutive days. For example, if a stock's value for a 5 day stretch was 36.07, 36.99, 36.21, 35.11 and 35.76, then the biggest change occured between days 3 and 4, with a loss of 1.1 points. Here is the input file format:

The first line of the input file will contain a single positive integer, *n*, representing the number of stocks to track. Each of the following *n* lines will contain data about one stock.

The first number on each of these lines will be a positive integer, *d*, representing the number of days for which stock data exists for that particular stock. This will be followed by *d* real numbers, all separated by spaces indicating the prices of that stock, in chronological order.

For each stock, print out a single line with the following format:

```
Stock #k: The biggest one day change was Y points.
```

where k is the stock number, starting with 1, and Y is a real number printed to two decimal places representing the largest change for that stock.

Sample Input File

```
2
5 36.07 36.99 36.21 35.11 35.76
4 31.27 30.33 30.57 32.12
```

Corresponding Output

```
Stock #1: The biggest one day change was 1.10 points.
Stock #2: The biggest one day change was 1.55 points.
```