Homework Five

A201/A597/I210
Spring Semester 2005

Due in writing at the end of the lecture on Thu, Mar 3*

Abstract

Write programs that solve the problems below as indicated. Programs should be submitted to OnCourse in the dropbox for Homework Five. A report including a very detailed explanation for each program describing the design and implementation as well as the rationale behind your solution and/or approach is to be submitted in writing at the time of the lecture on Thursday, March 3. You're encouraged to work in groups and discuss the problems but you need to write the programs and prepare the write-up all by yourself. The Computer Science Department[1] and the School of Informatics [2] clearly specify the rules of academic honesty and academic integrity, so please read the documents and make sure you understand them and comply with them.

Also posting solutions or major hints on the bulletin board is not allowed.

1 The Summary

This homework assignment is about using loops to (process and) summarize sequences of numbers. Your Lab Seven should be organized to help you get started with this assignment. Numbers, grouped in sequences, will be entered in your program as strings (lines of text). On each line numbers will be separated by spaces. It will be necessary to use a string tokenizer to break the strings into tokens (the numbers). The tokens will need to be parsed if they are to be used as numbers.

2 The Problems

2.1 Count the Number of Tokens

Write a program that reads lines of text and counts and reports the number of tokens on each line. The tokens don't have to be just numbers. Any

*OnCourse dropbox (for code) to stay open until 11:59pm on Friday, March 4.
[1] `http://www.cs.indiana.edu/Academics/integrity.html`
[2] `http://www.informatics.indiana.edu/courses/honesty.asp`

sequence of characters not separated by blank space counts as a token. Here's how my program behaves:

```
frilled.cs.indiana.edu%java One
enter> one two three 1 2 3
6 tokens on this line.
enter> I was here on Sunday.
5 tokens on this line.
enter> token token token token---by r.e.m.
5 tokens on this line.
enter> token --- token
3 tokens on this line.
enter> token--- token
2 tokens on this line.
enter>
0 tokens on this line.
enter> 1 2 3 4 5six
5 tokens on this line.
enter> done
Thank you for using this program.
frilled.cs.indiana.edu%
```

2.2 Calculate a Sum of Numbers

Write a program that reads lines of text from the user. Each line must contain zero or more numbers, separated by blank space. The program must report the sum of the numbers on each line. If there are n numbers on a line and numbers are denoted by x_i (with i starting at 1) then the sum your program calculates is

$$\sum_{i=1}^{n} x_i = x_1 + \ldots + x_{n-1} + x_n$$

It is the user's responsibility to ensure that the lines consist of only numbers. You may make use of the new `try/catch` feature that was presented in lecture, if you want, but it's not a requirement of this homework assignment in any way. Here's how my program works:

```
frilled.cs.indiana.edu%java Two
enter> 1 2 3
The sum is: 6
enter>
The sum is: 0
enter> 1 -1 2 -2 5 -2 -1 -3
The sum is: -1
enter> -1 3 -2
The sum is: 0
enter> 1 2 3 4 5 6 7 8 9 10
The sum is: 55
enter> done
frilled.cs.indiana.edu%
```

2.3 Calculate the Average

Write a program that combines the two programs above. The program reads lines of text from the user and reports the average of the numbers x_i on each line:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

It is the user's responsibility to ensure that only numbers appear on each line, just as before. The program reports the sum of the numbers and their count, together with their average (sum divided by count).

Here's my program in action:

```
frilled.cs.indiana.edu%java Three
enter> 1 2 3
  Sum is: 6
  Count is: 3
  Average is: 2.0
enter> 2 2 3
  Sum is: 7
  Count is: 3
  Average is: 2.3333333333333335
enter>
No numbers, no average.
enter> 1 1 1 -3
  Sum is: 0
  Count is: 4
  Average is: 0.0
enter> 12 3 45 678
  Sum is: 738
  Count is: 4
  Average is: 184.5
enter> done
frilled.cs.indiana.edu%
```

2.4 The Sum of Squares

Write a program that reads lines of text (containing numbers separated by blanks) from the user and reports the sum of the squares of the numbers, for each line. Using the notation from the previous problems, if x_i are the numbers on a line, the program reports and calculates:

$$\sum_{i=1}^{n} {x_i}^2 = x_1^2 + x_2^2 + \ldots + x_n^2$$

As before, if the user tokens that don't parse as numbers the behavior of the program is completely unspecified (that is, up to you). In the examples below I act as a good user (program-friendly user).

```
frilled.cs.indiana.edu%java Four
enter> 1 2 3
The sum is: 14
```

```
enter> -1 -1 -1
The sum is: 3
enter> -1 2 -2
The sum is: 9
enter> 12 34 5 678
The sum is: 461009
enter> done
frilled.cs.indiana.edu%
```

2.5 The Standard Deviation (I)

Use the expertise you have built in the previous programs to calculate the standard deviation for a sequence of numbers using the following formula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2}{n-1}}$$

Although intimidating, this formula is very simple: it says that you can calculate first $a = (\sum_{i=1}^{n} x_i)^2$, which is the square of the sum of the numbers on the line, then $b = \sum_{i=1}^{n} x_i^2$, that is, the sum of the squares of the numbers, and then plug these values into the original formula

$$\sigma = \sqrt{\frac{b - \frac{1}{n}a}{n-1}}$$

Here's my program, in action:

```
frilled.cs.indiana.edu%java Five
enter> 1 2 3
  Count is: 3
  Sum is: 6
  Sum of squares is: 14
  Standard deviation is: 1.0
enter> 4 4 4 1
  Count is: 4
  Sum is: 13
  Sum of squares is: 49
  Standard deviation is: 1.5
enter> 90 90 92 86
  Count is: 4
  Sum is: 358
  Sum of squares is: 32060
  Standard deviation is: 2.516611478423583
enter>
  Count is: 0
  Sum is: 0
  Sum of squares is: 0
  Standard deviation is: NaN
enter> done
Thanks for using this program.
frilled.cs.indiana.edu%
```

2.6 The Standard Deviation (II)

Rewrite the program above to use a different (more traditional) formula for the standard deviation:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}}$$

where

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

is the mean of the x_i values.

In this new case it will be necessary to calculate the result in two steps: start by creating a string tokenizer and use it to go through the values in the sequence, to calculate the average (mean) μ. Then create a new string tokenizer and go through the values in the sequence one more time, to calculate the squared deviations, and sum them up, to calculate the standard deviations. Here's my program in action:

```
frilled.cs.indiana.edu%java Six
enter> 1 2 3
  The average is: 2.0
  The sum of squared deviations is: 2.0
  The standard deviation is: 1.0
enter> 4 4 4 1
  The average is: 3.25
  The sum of squared deviations is: 6.75
  The standard deviation is: 1.5
enter> 90 90 92 86
  The average is: 89.5
  The sum of squared deviations is: 19.0
  The standard deviation is: 2.516611478423583
enter>
  The average is: NaN
  The sum of squared deviations is: 0.0
  The standard deviation is: -0.0
enter> done
frilled.cs.indiana.edu%
```

2.7 The Max

Write a program that accepts lines of text from the user, scans them and reports the largest number on each line. As before, it is the responsibility of the user to make sure the lines have only numbers on them. Here's how my program works:

```
frilled.cs.indiana.edu%java Seven
enter> 1 2 3
The largest number is: 3
enter> 3 2 1
The largest number is: 3
```

```
enter>
Empty line.
enter> 10
The largest number is: 10
enter> 10 2 3 4 5 11 9
The largest number is: 11
enter> done
frilled.cs.indiana.edu%
```

You may find the `StringTokenizer`'s `countTokens()` method useful here.

2.8 The Min

Repeat the program above except make it report the smallest number on each line. Here's how my program works:

```
frilled.cs.indiana.edu%java Eight
enter> 1 2 3
The smallest number is: 1
enter> -3
The smallest number is: -3
enter> 3 2 1
The smallest number is: 1
enter> 10
The smallest number is: 10
enter> 10 2 3 4 5 11 9
The smallest number is: 2
enter> 0 9 8 7 6 -3 23
The smallest number is: -3
enter> done
frilled.cs.indiana.edu%
```

2.9 The Simplified Vending Machine (SVM)

Write a program that reads coin denominations from the user, one per line, and keeps track of the total amount of money entered thus far. Acceptable denominations for coins are: `cent`, `nickel`, `dime`, `quarter`, `dollar`. Here's how my program works:

```
frilled.cs.indiana.edu%java Nine
enter> cent
Your balance is: 0.01
enter> nickel
Your balance is: 0.06
enter> dime
Your balance is: 0.16
enter> ddime
I don't understand ddime
Your balance is: 0.16
enter> dollar
Your balance is: 1.16
```

```
enter> dollar dollar
I don't understand dollar dollar
Your balance is: 1.16
enter>
I don't understand
Your balance is: 1.16
enter> quarter
Your balance is: 1.41
enter> quarter
Your balance is: 1.66
enter> dome
I don't understand dome
Your balance is: 1.66
enter> done
frilled.cs.indiana.edu%
```

2.10 The More Sophisticated SVM

Extend the previous program where the user can enter more than one coin keyword per line. Here's how my program works:

```
frilled.cs.indiana.edu%javac Ten.java
frilled.cs.indiana.edu%java Ten
enter> dime dime quarter
Your balance is: 0.45
enter> cent
Your balance is: 0.46
enter> cent cent dent
I don't understand dent
Your balance is: 0.48
enter> dime dime dime dollar dollar
Your balance is: 2.78
enter> done
frilled.cs.indiana.edu%
```