Homework Three

A201/A597/I210
Spring Semester 2005

Due in writing at the end of lecture on Feb. 15

Abstract

Write programs that solve the problems below as indicated. Programs should be submitted to OnCourse in the dropbox for Homework Three. 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 Tuesday, February 15, 2005. 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 is about using `while` loops to repeat a block of statements in a program. Don't forget that a `while` statement is just another statement, except it has a somewhat more sophisticated structure:

```
while (<condition>) {
  <statement>
  <statement>
  ...
}
```

The `while` statement is a *compound* statement (like the `if` statement and unlike the assignment statement, which is a *simple* statement). We encourage you to keep using curly braces to clearly delineate the body of the loop, that is, the block that needs to be executed (even though you can sometimes drop the curly braces, if the block is just one statement long). You can try to simplify your code only after you thoroughly master the basics. So this is what we will be working on in this assignment: the basics of using `while` statements in your programs.

[1] `http://www.cs.indiana.edu/Academics/integrity.html`
[2] `http://www.informatics.indiana.edu/courses/honesty.asp`

2 The Problems

2.1 Calculating the GPA

Write a program that uses a `while` loop to read any number of individual scores from the keyboard, one score per line. The program should disregard values outside the range $[0, 100]$. The user should indicate the end of the data input by entering the keyword `done`. The program should report how many scores were entered and what their average was. If the average is above 90 the user is congratulated.

2.2 A Menu of Songs

Write a program that uses a `while` loop to

- print a menu of songs (including an option to quit)
- read an option (an integer) from the user, then
- render the song for that option or quit, as indicated.

2.3 Addition Quiz

Write a program that uses a `while` loop to play the addition quiz from Homework Two with the user until the user types the word `done`. The program reports the current score after each step and prints a `"Thank you"` message at the end.

2.4 Paper, Rock, Scissors

Write a program that uses a `while` loop to play the Paper, Rock, Scissors game of Homework Two until the user types the word `done`. The program reports the current score after each step and prints a `"Thank you"` message at the end.

2.5 Clock

Same for the `Clock` problem of Homework Two: the clock is reported as a prompt. When the user hits enter the clock advances by one minute. When the user enters `done` the program prints a `"Thank you"` message and quits.

2.6 Digits

Write a program that uses a `a while` loop to read numbers from the user, one number per line. For every line the program should use a `while` loop to calculate the sum of the digits of that number. The program ends when the user enters the string `done`.

2.7 Reverse

Write a program that uses a `a while` loop to read lines from the user. For every line the program should use a `while` loop to print the line reversed. The program ends when the user enters the string `done`.

2.8 Square Root

Write a program that uses a `while` loop to implement the square root calculation of Homework One. The program should calculate new approximations while the square of the approximation is more than $0.000001 = 10^{-6}$ away from the number whose square root we're calculating.

2.9 Investment

Write a program that uses a `while` loop to implement the investment calculation of Homework One. The program should ask for an initial balance and an interest rate (both greater than zero). It should then calculate the yearly changes until the balance becomes twice the initial balance.

2.10 Counting

Write a program that uses a `while` loop to read lines from the user. For every line the user enters the program should report the number of vowels (lowercase or uppercase) in it. The program should also count the number of lines the user enters. When the user enters `done` the program ends.

3 More Examples and Hints

I decided to post some examples and hints *in case you need them*. Give yourself a few degrees of freedom and implement a reasonably similar program. Don't follow these hints *ad litteram* just because we gave them. But feel free to make use of them if you don't know how to get started.

3.1 Calculating the GPA

Here's how your program might work:

```
frilled.cs.indiana.edu%javac GPA.java
frilled.cs.indiana.edu%java GPA
Please enter a grade: 100
100.0 entered, thank you.
Please enter a grade: 98
98.0 entered, thank you.
Please enter a grade: 100
100.0 entered, thank you.
Please enter a grade: 90
90.0 entered, thank you.
Please enter a grade: done
```

```
4 scores, average is 97.0
Congratulations, the average is above 90.
frilled.cs.indiana.edu%java GPA
Please enter a grade: 80
80.0 entered, thank you.
Please enter a grade: -10
-10.0 is not a valid score. Disregarded.
Please enter a grade: 90
90.0 entered, thank you.
Please enter a grade: done
2 scores, average is 85.0
Average is not above 90, just letting you know.
frilled.cs.indiana.edu%java GPA
Please enter a grade: done
No scores, no average.
frilled.cs.indiana.edu%
```

And here are some hints: One way of solving this problem would be to follow the pattern in Lecture Notes Nine. Read strings (lines) from the user, and check for the termination keyword. If the user has not entered the word `done` you should assume that the user has entered a number. Because you have a string at this point you should convert to a double. Check if the number is in the range of interest and if it is not simply discard it, but also let the user know. If it is a score that you need to take into account, two actions are needed: increment by one the variable that helps you keep track of how many good scores you have seen thus far and add the good score to the sum of scores taken into account thus far. The addition quiz and the paper rock scissors problems in Homework Two make use of this technique. Obviously, these variables need to be defined very early in the program since they have to be available throughout the rest of the program. At the end of the program there are two possibilities: (a) either the user has not typed any number, so no average can be computed and in this case you should display a message, or (b) the user has typed at least one good score, in which case an average should be calculated (divide the sum by the number of good scores entered) and reported. When we report it we also check it against 90. If it is greater we congratulate the user.

3.2 A Menu of Songs

To simplify here's how your program should/could work:

```
frilled.cs.indiana.edu%javac Menu.java
frilled.cs.indiana.edu%java Menu
The menu is: 1, 2, 3, 4 (and done to quit):
Type something: 1
You have typed option 1
Type something: 4
You have typed option 4
Type something: nothing
Sorry, I don't understand option nothing
Type something: 10
```

```
Sorry, I don't understand option 10
Type something: 3
You have typed option 3
Type something: 6
Sorry, I don't understand option 6
Type something: 4
You have typed option 4
Type something: 3
You have typed option 3
Type something: 2
You have typed option 2
Type something: 1
You have typed option 1
Type something: done
frilled.cs.indiana.edu%
```

Hints on how to go about it: again Lecture Notes Nine (*The Conversation*) gives you the pattern you need. This time around you don't even need to convert the strings into numbers since you will not need them in any calculation. So I simply compared them against the strings that I knew they could be: `"1"`, `"2"` and so on. There is no need to print the songs, unless you want to.

3.3 The Addition Quiz

Here's how my program works:

```
frilled.cs.indiana.edu%javac Quiz.java
frilled.cs.indiana.edu%java Quiz
What is -35 + 18? 100
Your current score is: 0/1
What is 38 + -32? 6
Your current score is: 1/2
What is 38 + -33? 5
Your current score is: 2/3
What is 4 + -21? -17
Your current score is: 3/4
What is -11 + -20? 31
Your current score is: 3/5
What is -26 + -23? -49
Your current score is: 4/6
What is -33 + -33? done
Thank you for using this program.
frilled.cs.indiana.edu%java Quiz
What is 28 + 34? done
Thank you for using this program.
frilled.cs.indiana.edu%java Quiz
What is -43 + -6? nothing
Exception in thread "main" java.lang.NumberFormatException:
For input string: "nothing" at java.lang.NumberFormatException.[...]
frilled.cs.indiana.edu%
```

The same pattern of prompting the user for input, reading a line, checking to see if it means the end of our interaction with the user, or converting it into an answer, is what the problem is testing.

Solutions of the programs given in Homework Two contain the basic mechanism of generating the two random numbers, asking for an answer and reading it, comparing it with the sum of the two numbers that appear in the question and keeping a tally of good answers and a total number of questions asked thus far.

3.4 Paper-Rock-Scissors

Here's how my program works:

```
frilled.cs.indiana.edu%javac PRS.java
frilled.cs.indiana.edu%java PRS
The computer has chosen paper. What do you choose? rock
The computer has won. The score is now user 0 - computer 1
The computer has chosen scissors. What do you choose? paper
The computer has won. The score is now user 0 - computer 2
The computer has chosen paper. What do you choose? rock
The computer has won. The score is now user 0 - computer 3
The computer has chosen scissors. What do you choose? scissors
This is a tie. The score is still 0 - computer 3
The computer has chosen scissors. What do you choose? rock
The user has won. The score is now user 1 - computer 3
The computer has chosen rock. What do you choose? paper
The user has won. The score is now user 2 - computer 3
The computer has chosen rock. What do you choose? scissors
The computer has won. The score is now user 2 - computer 4
The computer has chosen paper. What do you choose? paper
This is a tie. The score is still 2 - computer 4
The computer has chosen rock. What do you choose? paper
The user has won. The score is now user 3 - computer 4
The computer has chosen paper. What do you choose? done
Thanks for using this program.
frilled.cs.indiana.edu%java PRS
The computer has chosen rock. What do you choose? nothing
Sorry, I don't understand nothing
The computer has chosen scissors. What do you choose? paper
The computer has won. The score is now user 0 - computer 1
The computer has chosen rock. What do you choose? scissors
The computer has won. The score is now user 0 - computer 2
The computer has chosen scissors. What do you choose? scissors
This is a tie. The score is still 0 - computer 2
The computer has chosen rock. What do you choose? nothing
Sorry, I don't understand nothing
The computer has chosen rock. What do you choose? done
Thanks for using this program.
frilled.cs.indiana.edu%
```

Be careful with this program, although the pattern is (again, very) simple: prompt the user, read input, and while the user does not appear to want to finish the game, process the user input, prompt the user for more input, read it and do it again.

What variables do you think you need, and what is the most difficult part of the program? You need to keep track of the user score and the computer's score (both start at 0) and you need to be able to convert random numbers into strings and the strings that the user enters into numbers, to compare with the computer choice. Since this has already been done in Homework Two you can simply use that part here.

3.5 Clock

Here's how my program works:

```
frilled.cs.indiana.edu%javac Clock.java
frilled.cs.indiana.edu%java Clock
Where do we start? 23:55
23:55>
23:56>
23:57>
23:58>
23:59>
00:00>
00:01>
00:02>
00:03>done
frilled.cs.indiana.edu%java Clock
Where do we start? 10:00
10:00>
10:01>done
frilled.cs.indiana.edu%java Clock
Where do we start? 09:58
09:58>
09:59>
10:00>
10:01>
10:02>done
frilled.cs.indiana.edu%
```

Please don't forget to decide ahead of time what variables you might need, and once inside the loop don't forget to ask the user for more input (otherwise, the loop might run forever). In my case I use two variables for the hour and the minute (as strings) and two variables for the hour and the minute (as integers). I also put together the clock's display as a string, being careful to pad with zeroes.

Well, it appears that I forgot to add a `Thank you!` ending to my program, but that shouldn't be hard to do. It amounts to a `println` just outside the loop.

3.6 Digits

Here's how my program works:

```
frilled.cs.indiana.edu%javac Digits.java
frilled.cs.indiana.edu%java Digits
Type a number: 12
The sum of the digits is: 3
Type a number: 101
The sum of the digits is: 2
Type a number: 1234
The sum of the digits is: 10
Type a number: 2098
The sum of the digits is: 19
Type a number: done
Thanks for using this program.
frilled.cs.indiana.edu%
```

Notice that for each line that you read you need to write a loop that goes through the string character by character. Start by initializing a variable (`sum`) with 0, and another one (`i`) also by 0, using `i` to move through the characters in the string, extract the digits as substrings of one character and parse them into actual digits (numbers) and sum them up, for reporting. You have to be careful to carefully increment `i` and carefully ask for more input from the user after each calculation, otherwise any of the two loops could run forever.

3.7 Reverse

Here's how my program works:

```
frilled.cs.indiana.edu%javac Reverse.java
frilled.cs.indiana.edu%java Reverse
Type something: hasta la vista baby
hasta la vista baby reversed is ybab atsiv al atsah
Type something: gniddik on
gniddik on reversed is no kidding
Type something: oh, yes!
oh, yes! reversed is !sey ,ho
Type something: say, ho!
say, ho! reversed is !oh ,yas
Type something: done
frilled.cs.indiana.edu%
```

This and the previous program are almost the same. Except in this one one starts with the index `i` at the end of the string (`line.length() - 1`, and subtracts `1` from `i` and prints the character at index `i` for as long as the index (which keeps going down) is still above or equal to `0` (zero).

3.8 Square Root

You should remember that the user enters the number (n) and the computer always makes the same initial guess: `1` (one). Following, the formula

$x = \frac{1}{2}(x + \frac{n}{x})$ is applied over and over until $|x^2 - n| < 10^{-6}$

Here's how my program works:

```
frilled.cs.indiana.edu%java Square
Please enter a positive number: 5
New guess: 1.0
New guess: 3.0
New guess: 2.3333333333333335
New guess: 2.238095238095238
New guess: 2.2360688956433634
2.236067977499978 squared is 5.000000000000843
frilled.cs.indiana.edu%java Square
Please enter a positive number: 141
New guess: 1.0
New guess: 71.0
New guess: 36.49295774647887
New guess: 20.178358456411956
New guess: 13.583021413252386
New guess: 11.9818139429309
New guess: 11.874824075819586
11.874342096819671 squared is 141.0000002323038
frilled.cs.indiana.edu%java Square
Please enter a positive number: 300
New guess: 1.0
New guess: 150.5
New guess: 76.24667774086379
New guess: 40.09063770159931
New guess: 23.78684077710257
New guess: 18.199428042345787
New guess: 17.341731278582593
New guess: 17.320521062406588
17.320508075693642 squared is 300.00000000016865
frilled.cs.indiana.edu%
```

3.9 Investment

The key here is to keep track of the changing balance while still remembering the original amount. Thus the `while` loop will compare the original balance with the current one and stop when the current one is at least twice as big. Here's how my program works:

```
frilled.cs.indiana.edu%javac Investment.java
frilled.cs.indiana.edu%java Investment
Initial balance: 100
Interest rate (a number between 0 and 1): 0.03
1. The balance becomes: 103.0, the ratio is 1.03
2. The balance becomes: 106.09, the ratio is 1.0609
3. The balance becomes: 109.2727, the ratio is 1.092727
4. The balance becomes: 112.550881, the ratio is 1.1255088100000001
5. The balance becomes: 115.92740743, the ratio is 1.1592740743
6. The balance becomes: 119.4052296529, the ratio is 1.1940522965290001
```

```
7. The balance becomes: 122.987386542487, the ratio is 1.22987386542487
8. The balance becomes: 126.67700813876162, the ratio is 1.2667700813876162
9. The balance becomes: 130.47731838292447, the ratio is 1.3047731838292447
10. The balance becomes: 134.39163793441222, the ratio is 1.3439163793441222
11. The balance becomes: 138.4233870724446, the ratio is 1.384233870724446
12. The balance becomes: 142.57608868461793, the ratio is 1.4257608868461793
13. The balance becomes: 146.85337134515646, the ratio is 1.4685337134515646
14. The balance becomes: 151.25897248551115, the ratio is 1.5125897248551115
15. The balance becomes: 155.7967416600765, the ratio is 1.5579674166007649
16. The balance becomes: 160.4706439098788, the ratio is 1.604706439098788
17. The balance becomes: 165.28476322717515, the ratio is 1.6528476322717516
18. The balance becomes: 170.2433061239904, the ratio is 1.7024330612399041
19. The balance becomes: 175.3506053077101, the ratio is 1.7535060530771012
20. The balance becomes: 180.61112346694142, the ratio is 1.8061112346694141
21. The balance becomes: 186.02945717094966, the ratio is 1.8602945717094965
22. The balance becomes: 191.61034088607815, the ratio is 1.9161034088607816
23. The balance becomes: 197.3586511126605, the ratio is 1.973586511126605
24. The balance becomes: 203.27941064604033, the ratio is 2.032794106460403
frilled.cs.indiana.edu%java Investment
Initial balance: 1000
Interest rate (a number between 0 and 1): 0.3
1. The balance becomes: 1300.0, the ratio is 1.3
2. The balance becomes: 1690.0, the ratio is 1.69
3. The balance becomes: 2197.0, the ratio is 2.197
frilled.cs.indiana.edu%
```

3.10 Counting

In this program you will need to keep three tallies: one will be with respect to the number of lines, and a second one (for each line) will be initialized with 0 (zero) and will traverse the string from left to right, character by character, looking for vowels (`'a'`, `'e'`, `'i'`, `'o'`, `'u'`, uppercase or lowercase). Once a vowel is found a third variable (that always starts at 0 for each line) is incremented. It will be used to report the number of vowels on each line. Here's how my program works:

```
frilled.cs.indiana.edu%java Vowels
Type something: what do you want me to type?
1. what do you want me to type? has 8 vowels in it.
Type something: vowels are my favourite
2. vowels are my favourite has 9 vowels in it.
Type something: favorite
3. favorite has 4 vowels in it.
Type something: who are you?
4. who are you?  has 5 vowels in it.
Type something: asked the caterpillar
5. asked the caterpillar has 7 vowels in it.
Type something: done
frilled.cs.indiana.edu%
```