Make-Up Problems (Bonus) Batch One

A201/A597/I210
Spring Semester 2005

The only way to make up and review at the same time.

Abstract

Take advantage of this make-up if any of your first three assignments was less than perfect. This is also a wonderful opportunity to get more practice before the first Midterm Exam in RH 100, 7-9pm, on W (2/16).

Write programs that solve the problems below as indicated. Programs should be submitted to OnCourse in the dropbox marked Bonus Problems. 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 is a cummulative assignment only it counts for any part of Homework Assignments 1, 2, or 3 (in case you didn't do as well as you wanted on any of them). Work as many problems as you want, and turn them in. Each problem is about 7-9 points. Solving them all is also encouraged, because it's good practice for the exam. A 90 on an assignment is an A.

2 The Problems

2.1 Acres

One acre of land is equivalent to 43,560 square feet. Write a program that calculates the number of acres in a tract of land whose area is indicated by

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

the user. (Program asks the user for a value, user enters value, program calculates the conversion and reports the calculated value).

Here's how my program works:

```
frilled.cs.indiana.edu%javac Acres.java
frilled.cs.indiana.edu%java Acres
What's the area (in acres): 1
1.0 acres amount to 43560.0 square feet
frilled.cs.indiana.edu%java Acres
What's the area (in acres): 1.41
1.41 acres amount to 61419.6 square feet
frilled.cs.indiana.edu%java Acres
What's the area (in acres): 0.001
0.0010 acres amount to 43.56 square feet
frilled.cs.indiana.edu%
```

2.2 Averages

Write a program that asks the user to enter a number per line and prints back (after every line) the current average of all the numbers entered thus far. Program stops when the user enters `done`, alone, on the line.

My programs works like this:

```
frilled.cs.indiana.edu%javac Averages.java
frilled.cs.indiana.edu%java Averages
Enter number: 34
1. Average is: 34.0
Enter number: 10
2. Average is: 22.0
Enter number: 120
3. Average is: 54.666666666666664
Enter number: -10
4. Average is: 38.5
Enter number: done
frilled.cs.indiana.edu%
```

2.3 Calendar

Write a program that simulates the passing of days in a non-leap year (essentially simulating the calendar). You give the program a date and a number of days and it prints back the days, one by one.

This is how my program works:

```
frilled.cs.indiana.edu%java Calendar
Type the starting date (dd/mm): 02/26
Type the number of days: 5
02/27
02/28
03/01
03/02
03/03
```

```
frilled.cs.indiana.edu%java Calendar
Type the starting date (dd/mm): 12/29
Type the number of days: 7
12/30
12/31
01/01
01/02
01/03
01/04
01/05
frilled.cs.indiana.edu%java Calendar
Type the starting date (dd/mm): 09/29
Type the number of days: 10
09/30
10/01
10/02
10/03
10/04
10/05
10/06
10/07
10/08
10/09
frilled.cs.indiana.edu%
```

2.4 Matching Strings

Write a program that reads two lines of text and reports the number of places where the two lines coincide. For example: if manitoba and baritone are the two given words, they only match in four places, that is, in the second, fourth, fifth and sixth positions (as indicated). (Note: the strings don't have to have the same length!)

My program works like this:

```
frilled.cs.indiana.edu%java Match
Type the first string: manitoba
Type the second string: baritone
manitoba and
baritone are identical in 4 positions:  1 3 4 5
frilled.cs.indiana.edu%java Match
Type the first string: anthropomorphic
Type the second string: anteater
anthropomorphic and
anteater are identical in 3 positions:  0 1 2
frilled.cs.indiana.edu%
```

2.5 Calculations

Write a program that calculates

$$\frac{1}{n} + \frac{2}{n-1} + \frac{3}{n-2} + \ldots + \frac{n-1}{2} + \frac{n}{1}$$

for $n = 30$. Ideally program asks user for of n then calculates the sum.

Here's how my program works:

```
frilled.cs.indiana.edu%java Calculations
Type the number: 30
The sum is: 93.84460105853213
frilled.cs.indiana.edu%java Calculations
Type the number: 2
The sum is: 2.5
frilled.cs.indiana.edu%java Calculations
Type the number: 10
The sum is: 22.218650793650795
frilled.cs.indiana.edu%
```

2.6 Characters

Write a program that counts the number of times a character occurs in a string. Here's how my program works:

```
frilled.cs.indiana.edu%javac Count.java
frilled.cs.indiana.edu%java Count
Type a string: concatenation
Now type a single character: o
o occurs 2 times in concatenation
frilled.cs.indiana.edu%java Count
Type a string: concatenation
Now type a single character: n
n occurs 3 times in concatenation
frilled.cs.indiana.edu%java Count
Type a string: baritone
Now type a single character: x
x occurs 0 times in baritone
frilled.cs.indiana.edu%
```

2.7 Table of Temperatures

Write a program that displays a table of the centigrade temperatures $x_0 = 0$ through $x_1 = 20$ and their Fahrenheit equivalents. Ideally the program asks the user for the two values, makes sure that $x_0 < x_1$ and displays the table of temperatures for $x_0, x_0 + 1, \ldots, x_1 - 1, x_1$.

Here's how my program works:

```
frilled.cs.indiana.edu%java Temps
Please type x0: 0
Please type x1: 20
0.0C is 32.0F
1.0C is 33.8F
2.0C is 35.6F
3.0C is 37.4F
4.0C is 39.2F
5.0C is 41.0F
```

```
6.0C is 42.8F
7.0C is 44.6F
8.0C is 46.4F
9.0C is 48.2F
10.0C is 50.0F
11.0C is 51.8F
12.0C is 53.6F
13.0C is 55.400000000000006F
14.0C is 57.2F
15.0C is 59.0F
16.0C is 60.8F
17.0C is 62.6F
18.0C is 64.4F
19.0C is 66.2F
20.0C is 68.0F
frilled.cs.indiana.edu%java Temps
Please type x0: 20
Please type x1: 0
Sorry, 20.0 is not smaller than 0.0
frilled.cs.indiana.edu%java Temps
Please type x0: 30
Please type x1: 40
30.0C is 86.0F
31.0C is 87.80000000000001F
32.0C is 89.6F
33.0C is 91.4F
34.0C is 93.2F
35.0C is 95.0F
36.0C is 96.8F
37.0C is 98.60000000000001F
38.0C is 100.4F
39.0C is 102.2F
40.0C is 104.0F
frilled.cs.indiana.edu%
```

2.8 Stenograph

Write a program that "removes" all vowels from a string of characters. Essentially you need to traverse the string and create another one that only contains the consonants. Here's how my program works:

```
frilled.cs.indiana.edu%javac Steno.java
frilled.cs.indiana.edu%java Steno
Please enter a string: this is a random sentence
The string without vowels is: (ths s  rndm sntnc)
frilled.cs.indiana.edu%java Steno
Please enter a string: good evening and welcome to minneapolis
The string without vowels is: (gd vnng nd wlcm t mnnpls)
frilled.cs.indiana.edu%
```

2.9 Parens

Write a program that receives a string and rewrites it with parens around each vowel. Here's how my program works:

```
frilled.cs.indiana.edu%javac Parens.java
frilled.cs.indiana.edu%java Parens
Type a string: this is a longer string
th(i)s (i)s (a) l(o)ng(e)r str(i)ng
frilled.cs.indiana.edu%java Parens
Type a string: nothing
n(o)th(i)ng
frilled.cs.indiana.edu%
```

2.10 Converter

Write a program that converts between binary and decimal representations of integers. In other words if a binary number (which is simply a string of 0's and 1's) is given

$$(a_n a_{n-1} \dots a_3 a_2 a_1 a_0)_2$$

we need to calculate

$$m = a_0 + 2a_1 + 2^2 a_2 + \dots 2^n a_n = \sum_{k=0}^{n} a_k 2^k$$

Thus $1001_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2 + 1 \cdot 2^0 = 8 + 1 = 9$.

This is how my program works:

```
frilled.cs.indiana.edu%javac Converter.java
frilled.cs.indiana.edu%java Converter
Type the binary number: 1001
1001 in base 2 is 9 in base 10
frilled.cs.indiana.edu%java Converter
Type the binary number: 10101
10101 in base 2 is 21 in base 10
frilled.cs.indiana.edu%java Converter
Type the binary number: 111
111 in base 2 is 7 in base 10
frilled.cs.indiana.edu%java Converter
Type the binary number: 10011
10011 in base 2 is 19 in base 10
frilled.cs.indiana.edu%
```

2.11 Commas

Write a program that places commas in integers. By that I mean to add commas such that digits appear in groups of 3 from right to left separated by commas. To give an example here's how my program works:

```
frilled.cs.indiana.edu%javac Commas.java
frilled.cs.indiana.edu%java Commas
Type the number: 1234
The converted string is: 1,234
frilled.cs.indiana.edu%java Commas
Type the number: 123
The converted string is: 123
frilled.cs.indiana.edu%java Commas
Type the number: 1234567890
The converted string is: 1,234,567,890
frilled.cs.indiana.edu%java Commas
Type the number: it doesn't matter whether it's a number or not
The converted string is: i,t d,oes,n't, ma,tte,r w,het,her, it,'s ,a n,umb,er ,or ,not
frilled.cs.indiana.edu%java Commas
Type the number: abcdefghijklm
The converted string is: a,bcd,efg,hij,klm
frilled.cs.indiana.edu%java Commas
Type the number:
The converted string is:  ,   ,   ,
frilled.cs.indiana.edu%
```

So, as you can see, I treat the input as a string of characters. Truly, this is a program you need to approach with care. You would likely need to traverse the string from the end and count the characters you are visiting as you are placing them into the result (which amounts to a copy of the input string). When the position of the character is a multiple of 3 (that is, when `n % 3 == 0`) we need to place a comma in the result after the character. In all other cases we need to place the character without a comma.

2.12 Simple Program

Write a program that asks the user to enter a number. If the user enters the valid number (positive integer) the program prints the sum of the cubes of its digits and terminates. Otherwise the program keeps asking for a number until the user provides a number.

Let's use this problem to introduce a new Java feature. Congratulations to Andrei Patrulescu (working with Chris Talarico) for almost discovering this feature on their own, in their work for Assignment Three. Here's the basic structure of the program:

```
frilled.cs.indiana.edu%javac Cubes.java
frilled.cs.indiana.edu%java Cubes
Please enter a number: abc
Sorry, abc was not a valid number.
Please enter another number: 78.5
Sorry, 78.5 was not a valid number.
Please enter another number: -100
Sorry, -100 was not a valid number.
Please enter another number: 10a
Sorry, 10a was not a valid number.
```

```
Please enter another number: 12
Very good, 12 is a positive integer which we can use.
frilled.cs.indiana.edu%
```

This is the code for the program above:

```
class Cubes {
  public static void main(String[] args) {
    ConsoleReader c = new ConsoleReader(System.in);
    System.out.print("Please enter a number: ");
    String line = c.readLine();
    int n;
    try {
      n = Integer.parseInt(line);
    } catch (Exception e) {
      n = -1;
    }
    while (n < 0) {
      System.out.println("Sorry, " + line + " was not a valid number.");
      System.out.print("Please enter another number: ");
      line = c.readLine();
      try {
        n = Integer.parseInt(line);
      } catch (Exception e) {
        n = -1;
      }
    }
    System.out.println("Very good, " + n + " is a positive integer which we can use.");
  }
}
```

The highlighted section allows the Java programmer to catch any unusual situation and indicate an alternative to it. In this case we either succeed parsing or we don't. When we don't, we set the value of `n` to `-1`. Since we recognize that value as an unacceptable value we keep asking for a new positive integer until we receive it. Now it's a simple matter for you to write your program. Here's how mine works:

```
frilled.cs.indiana.edu%java Cubes
Please enter a number: -5
Sorry, -5 was not a valid number.
Please enter another number: 3.14
Sorry, 3.14 was not a valid number.
Please enter another number: abc
Sorry, abc was not a valid number.
Please enter another number: 10u
Sorry, 10u was not a valid number.
Please enter another number: 7
Very good, 7 is a positive integer which we can use.
The sum of the first 7 cubes is: 2401
frilled.cs.indiana.edu%
```

2.13 ASCII codes

Write a program that receives an ASCII numeric code and prints the character associated with it. Extend the program so it prints all the characters that have codes in a certain numeric range that is given.

Here's how my extended program works:

```
frilled.cs.indiana.edu%java ASCII
Start of range: 30
End of range of interest: 40
The character with code 30 is
The character with code 31 is
The character with code 32 is
The character with code 33 is !
The character with code 34 is "
The character with code 35 is #
The character with code 36 is $
The character with code 37 is %
The character with code 38 is &
The character with code 39 is ’
frilled.cs.indiana.edu%java ASCII
Start of range: 40
End of range of interest: 70
The character with code 40 is (
The character with code 41 is )
The character with code 42 is *
The character with code 43 is +
The character with code 44 is ,
The character with code 45 is -
The character with code 46 is .
The character with code 47 is /
The character with code 48 is 0
The character with code 49 is 1
The character with code 50 is 2
The character with code 51 is 3
The character with code 52 is 4
The character with code 53 is 5
The character with code 54 is 6
The character with code 55 is 7
The character with code 56 is 8
The character with code 57 is 9
The character with code 58 is :
The character with code 59 is ;
The character with code 60 is <
The character with code 61 is =
The character with code 62 is >
The character with code 63 is ?
The character with code 64 is @
The character with code 65 is A
The character with code 66 is B
The character with code 67 is C
```

```
The character with code 68 is D
The character with code 69 is E
frilled.cs.indiana.edu%
```

2.14 Eggy-Peggy

Write an Eggy-Peggy kind of program. Given a string, convert it to a new string by placing `egg` in front of every vowel. Here are some examples:

```
frilled.cs.indiana.edu%java EggyPeggy
Type a string: I love Java.
eggI leggovegge Jeggavegga.
Type a string: Hi.
Heggi.
Type a string: Convoluted.
Ceggonveggolegguteggedg.
Type a string: done
Theggank yeggoeggu.
frilled.cs.indiana.edu%
```

Not too bad.

2.15 Easy as Pi

Approximate `Math.PI` by calculating:

$$4 \cdot (1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \ldots)$$

Find out how many terms you need to go through to be 0.01 away (or less) from the value of `Math.PI` (which you can print if you want).

```
frilled.cs.indiana.edu%java Pi
Adding 1 * 4 / 1, sum becomes 4.0
Adding -1 * 4 / 3, sum becomes 2.666666666666667
Adding 1 * 4 / 5, sum becomes 3.466666666666667
Adding -1 * 4 / 7, sum becomes 2.8952380952380956
[...]
Adding -1 * 4 / 767, sum becomes 3.13898849133825
Adding 1 * 4 / 769, sum becomes 3.1441900518063903
Adding -1 * 4 / 771, sum becomes 3.1390019843615136
Thanks for using this program.
```

2.16 Stability

For

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n}$$

calculate the sum from the left and from the right and compare the results for $n = 50000$.

```
frilled.cs.indiana.edu%java Stability
From the left: 11.397003949278504
From the right: 11.397003949278519
The difference: -1.4210854715202004E-14
frilled.cs.indiana.edu%
```

2.17 Exponents

Find the smallest $n > 0$ such that 2^n is bigger than $10000n^2$

```
frilled.cs.indiana.edu%java Expo
For n = 1: 2.0 <= 10000
For n = 2: 4.0 <= 40000
For n = 3: 8.0 <= 90000
For n = 4: 16.0 <= 160000
For n = 5: 32.0 <= 250000
For n = 6: 64.0 <= 360000
For n = 7: 128.0 <= 490000
For n = 8: 256.0 <= 640000
For n = 9: 512.0 <= 810000
For n = 10: 1024.0 <= 1000000
For n = 11: 2048.0 <= 1210000
For n = 12: 4096.0 <= 1440000
For n = 13: 8192.0 <= 1690000
For n = 14: 16384.0 <= 1960000
For n = 15: 32768.0 <= 2250000
For n = 16: 65536.0 <= 2560000
For n = 17: 131072.0 <= 2890000
For n = 18: 262144.0 <= 3240000
For n = 19: 524288.0 <= 3610000
For n = 20: 1048576.0 <= 4000000
For n = 21: 2097152.0 <= 4410000
For n = 22: 4194304.0 <= 4840000
For n = 23: 8388608.0 > 5290000
frilled.cs.indiana.edu%
```

2.18 Permutations

Write a program that creates circular permutations of a given string.

```
frilled.cs.indiana.edu%java Permutations
Please enter a string: adrian
1. nadria
2. anadri
3. ianadr
4. rianad
5. driana
6. adrian
frilled.cs.indiana.edu%java Permutations
Please enter a string: abc
1. cab
```

```
2. bca
3. abc
frilled.cs.indiana.edu%java Permutations
Please enter a string: wonderful
1. lwonderfu
2. ulwonderf
3. fulwonder
4. rfulwonde
5. erfulwond
6. derfulwon
7. nderfulwo
8. onderfulw
9. wonderful
frilled.cs.indiana.edu%
```

This is simpler than the `Swap` of the first homework assignment. If the string is `abc` I need to take the first letter `(a)` and the remaining part of the string `(bc)` and create `(bc)(a)` (without the parens, which are there just to clarify what happens). Print, repeat, print, repeat, for as many characters as the string has.