ASSIGNMENT 3

COMP-202B, Winter 2008, All Sections

Due: Wednesday, February 20, 2008 (23:55)

You **MUST** do this assignment individually and, unless otherwise specified, you **MUST** follow all the general instructions and regulations for assignments.

Graders have the discretion to impose penalties to students who deviate from the general instructions and regulations; these penalties will take the form of deductions from the marks allocated for respect of instructions and regulations.

Part 1, Question 1:	0 points
Part 2, Question 1:	50 points
Part 2, Question 2:	50 points
	100 points total

Part 1 (0 points)

Do **NOT** *hand in this part, as it will not be graded. However, doing this exercise might help you to do the second part of the assignment (that will be graded). If you have difficulties with the questions of Part 1, then we suggest you go to one of the office hours; the TA can help you and work with you through the warm-up questions.*

Warm-up Question 1 (0 points)

Write a program that prompts the user to input a string. Your program should then analyze the characters of the word, and output the total number of letters, the total number of vowels (do not count 'y' as a vowel), and what percentage of the letters in the string are vowels.

For example, if the user inputs `"May the force be with you!"`, then your program should output `"8 of the 20 letters (40.00%) are vowels."`

Part 2 (50 + 50 = 100 points)

The questions in this part of the assignment will be graded.

Question 1 (50 points)

Write a program *BinaryConverter* that prompts the user to input a string. The string represents a number in binary format. Your program should output the number converted into decimal format. You can safely assume that the user will input a correct string (i.e. only use '0's and '1's), and that the user either inputs 8 bits, 16 bits or 32 bits.

Here's a brief reminder on how to convert a number represented in binary format to the corresponding number represented in decimal format. The first digit from the right represents the units (2^0), the second digit from the right the 2's (2^1), the third digit from the right represents the 4's (2^2), the fourth digit the 8's (2^3) and so on. This is just like the decimal format, where the first digit from the right represents the units (10^0), the second digit from the right the 10's (10^1), the next one the 100's (10^2), etc.

For example, the number `00110110` equals $0 * 2^0 + 1 * 2^1 + 1 * 2^2 + 0 * 2^3 + 1 * 2^4 + 1 * 2^5 = 54$.

For this question, *negative* numbers are represented using the *2's complement representation*. If the left-most bit is '1', then the number should be interpreted as a negative number. But how is the number encoded? In short, the 2's complement is the binary number you would have to add to the positive number of the same magnitude to get a result of zero (if you ignore the overflow). The 2's complement of a positive number can be calculated as follows: invert the number (i.e. change all '0's to '1's and all '1's to '0's) and then add 1.

For example, the 8-bit representation of the number 17 is `00010001`. The 8-bit 2's complement of 17 (representing the number -17) is equal to `11101111`. The 8-bit representation of the number 1 is `00000001`. The representation of -1 is `11111111`. Finally, using the 2's complement representation, there is one number that requires special treatment. The number in which the first digit is a 1 and all other digits are 0 is special, because it is it's own 2's complement. For example, the 2's complement of `10000000` is `10000000`. Per definition, this number is equal to $-(2^{n-1})$, where n is equal to the number of digits used in the binary number.

Please submit the file `BinaryConverter.java` on WebCT. Note that for this question you are *not* allowed to use any method provided by the Java class library that implements binary conversion for you. An example of such a forbidden method is `parseInt()` provided by the `Integer` class.

Question 2 (50 points)

Once a year, Prof. Philip Carl Salzman from the Anthropology department of McGill University is carrying out excavations in Syria, in order to find the oldest traces of prehistoric human activity in the Middle-East. The excavations reveal many different animal and human bones, and a lot of flints ("silex" in French). A flint is an oriented prehistoric stone implement (tool), that prehistoric men used for different activities, e.g. cutting, scraping, boring, hunting.

Flints can be classified according to their morphology.

- If a flint's height exceeds 2 times its width, it is called a *blade*. If a blade's height is smaller than 5 cm and its width smaller than 1.3 cm, it is also called a *bladelett*. (To clarify the situation, all bladeletts are also blades!)
- The other flints are called flakes. Among flakes, we distinguish between *elongated* flakes, whose height exceeds their width, and *wide* flakes. (To clarify the situation, flakes are either elongated or wide!)

The flints found in El Kowm (a village in central Syria) last year have been measured, and the height and width of each flint in *cm* has been written to a file. The file therefore contains an even number of real numbers. The numbers at odd positions in the file represent the height, the numbers at even positions represent the width of a flint. There are no data errors or characters in the file, because it was produced by a machine. The size of the file, i.e. how many flints have been measured, is unknown. However, after the measurements of the last flint, the file ends with the number 0.0[1].

[1] Obviously, no flint can have a height or width of 0.0, so there can be no confusion between a flint measurement and the last number in the file.

Write a program *FlintAnalyzer* which first prompts the user for the name of the file to be analyzed. The program then reads the height and width of all flints from the file and analyzes their morphology. Once all measurements have been processed, the total number of flints and the number of flints in each category and their percentage must be displayed on the screen. The output generated by your program should look like this (percentage output should be rounded to 2 digits after the decimal point):

```
Please enter the name of the file to be processed: flints.txt
Total number of flints processed: 500
- 150 blades (30.00%), 50 (10.00%) of which are bladeletts
- 350 flakes (70.00%), of which 100 (20.00%) are elongated, and 250 (50.00%) are wide
```

Hint: You can use the Scanner class to read numbers from a file instead of the keyboard by instantiating it as follows: `Scanner scan = new Scanner(new File(filename))`. In order to make this work, you have to import the package *java.io* by writing `import java.io.*` at the top of your code, and add the words `throws FileNotFoundException` to the declaration of your `main` method (so that it looks like this: `public static void main (String[] args) throws FileNotFoundException`.

Please submit the file `FlintAnalyzer.java` on WebCT.