Programming Assignment 2
TCSS 142 Spring 2015
Due: Friday, June 5, 10:50 a.m.

Given the problem statement below, complete the following

- ✓ Program that is organized into multiple functions and runs correctly including (100 total pts)
  - Coding style (meaningful identifiers, indentation, etc.) (10 pts)
  - Comments
    - Function headers with pre and postconditions (10 pts)
    - Inline comments explaining code logic (5 pts)
    - Header with your name and course info at the top of the file

  A program must run in order to receive credit for this assignment. Modular design is an integral component of this project. Programs that do not use proper modularization will not receive a grade higher than 40 pts all together, even if a program runs perfectly and the documentation is correct and thorough.

Extra credit :

- ✓ Detailed test plan (extra credit 10 pts)

You are NOT allowed to help one another with this program or use somebody else's code – check the rules listed on the syllabus. However, you may seek help from CSS mentors, TAs, and instructors.

Problem statement

For this program, you are to design and implement a calculator for positive integers that uses John Napier's location arithmetic (explained below). The user of your program will enter two numbers, using Napier's notation, and an arithmetic operator. You program is to translate these numerals into Hindu-Arabic decimal numbers, display them, perform desired operations, and finally print the result - in both abbreviated Napier's notation and in Hindu-Arabic decimal notation. The arithmetic operators that your program should recognize are +, -, *, and /. These operators are used to perform Python operations of <u>integer</u> addition, subtraction, multiplication, and division. The program is to be repeated as many times as the user wishes.

You may <u>not</u> use Python constructs that have not been discussed in class so far.

Input

An input to your program will be interactive. Napier used letters to represent successive powers of 2, where a = 2^0, b = 2^1, c = 2^2, d = 2^3, … z = 2^{25}.

When converting to a decimal numeral you simply add the digits:

abdgkl = $2^0 + 2^1 + 2^3 + 2^4 + 2^6 + 2^{10} + 2^{11} = 1 + 2 + 8 + 64 + 1024 + 2048 = 3147$

His location numerals used sign-value notation, in which a number is simply the sum of its digits, where order does not matter, and digits can be repeated, e.g.

abc = cba = bca

abbc = acc = ad

After each number or an operator is entered, you are to make sure that no illegal characters or arithmetic operators were entered. For numbers, an illegal character is anything that is not a letter. For operators, an illegal character is anything other than one of the four operator characters listed above. When an invalid input occurs, the program is not to crash or exit, but to ask the user for the given value again – until the valid values are entered.

Output
In addition to printing the decimal values of the two numbers entered by the user, you need to print the result of the operation in both notations. When printing in Napier's notation, make sure you do not repeat letters where a single letter would do, e.g. instead of abbc, you should print ad. In addition, if the result of your calculation is a 0, the equivalent string would be an empty string which we will not be able to see on the screen, so in such a case print a set of quotes, e.g. "". If the result of your calculations is a negative number, print the number with a minus sign in front, e.g. –abc.

To represent a given decimal number as a location numeral, express it as a sum of powers of two and then replace each power of two by its corresponding digit (letter). For example, when converting from a decimal number:

$87 = 1 + 2 + 4 + 16 + 64 = 2^0 + 2^1 + 2^2 + 2^4 + 2^6 =$ abceg

You may find standard algorithms for converting from decimal to binary useful to achieve this purpose. To convert from a base-10 integer to its base-2 (binary) equivalent, the number is divided by two, and the remainder serves as the least-significant bit. The (integer) result is again divided by two, its remainder is the next least significant bit. This process repeats until the quotient becomes zero. Example:

13 / 2 = 6 R 1
 6 / 2 = 3 R 0
 3 / 2 = 1 R 1
 1 / 2 = 0 R 1

So 13 in decimal is 1101 in binary which basically tells you to look at letters corresponding to where 1s occurred in the bit stream, which in this case means positions 0, 2, and 3 = acd

Sample run

```
Enter Napier's number: -ab2
Something is wrong. Try again.
Enter Napier's number: abc
The first number is: 7
Enter Napier's number: he lo
Something is wrong. Try again.
Enter Napier's number: hello
The second number is 20,624
Enter the desired arithmetic operation: #
Something is wrong. Try again.
Enter the desired arithmetic operation: -
The result is -20,617 or -adhmo
Do you want to repeat the program? Enter y for yes, n for no:
```

Program Submission

If you want your assignment to be graded, it has to be compatible with our platform, namely Python3.4.1
The source code is to be called yournetid_project2.py

On or before the due date, use the link posted in Canvas next to Programming Assignment 2 to submit your code and the test plan, if any. Make sure you know how to do that before the due date since late assignments will not be accepted. Valid documentation file formats are: doc, docx, rtf, pdf.