CS 361 – Homework #1 – Creating Expressions – Due September 26, 2018

Write a computer program in either Java or Python that obtains six input numbers from the user. The goal of the program is to derive an arithmetic expression using some or all of the 6 input values, combining them using arithmetical operations chosen from +, -, *, /, so that the result equals a three-digit number (100-999). For example, if the 6 input values are 1, 2, 6, 9, 75 and 100, then one possible legal expression is 75 + 1 + 9 * (100 – 6 / 2), and this equals the 3-digit number 949. Your program will attempt to find a legal expression for all three-digit numbers.

Some stipulations and hints:

a. Your program will be tested in a Linux environment, e.g. the server `cs.furman.edu`. You need to make sure that your program will compile and run in this environment. If you are using Java, then your program should compile with the command `javac *.java`. Do not create a Java package.
b. Your program should adhere to good programming style. Do not indent excessively. Limit source lines to 80 columns. There should be a comment at the top of each source file, and above each function. Other comments in the code should indicate the important choices you are making in your algorithm, and indicate to the reader "where am I" in the code. Comments should also take note of assumptions that the program is making, and what steps in the algorithm are being accomplished. If your program does not exhibit acceptable programming style, it will not be evaluated.
c. All input will come from the command line. If you are using Java, then your main method must be in Driver.java, and the command to run your program will be of this form:
```
java Driver <six input values>
For example:  java Driver 9 6 2 1 75 100
```
If you are using Python, then call your source file numbers.py. The command to run your program will be of this form:
```
python numbers.py <six input values>
For example:  python numbers.py 9 6 2 1 75 100
```
d. Your program should attempt to find a solution for every three-digit number, of which there are 900: from 100 to 999. It is only necessary to find one solution for each three-digit number.
e. Give your program a 30-second time limit. In other words, your program should halt after about 30 seconds. At the end, your program should print the number of three-digit numbers for which a solution was found, a list showing each solution in ascending order of value from 100 to 999, as well as the total number of candidate expressions that it considered during the 30 seconds. See the example I/O given later.
f. You do not need to represent an arithmetic expression as a tree. Instead, you may find it convenient to represent it instead in postfix notation, and evaluate it using a stack.
g. The result of each operation in your output expression must be a positive integer. This will help eliminate some redundant answers.
h. You do not need to use all 6 input numbers in the expression. The smallest legal expression will contain two numbers and one operator.
i. The output must be expressed using postfix notation. Print one space between each token in the expression.
j. Your program does not need to intelligently "figure out" how to create the expression the way a human being would. In a nutshell, you may use trial and error. You need to consider:

- The ways the expression can be grouped
- The possible choices for the operators
- The ways to permute the input values inside the expression.

The combined number of permutations may be prohibitive. Billions of expressions are possible. So, I recommend generating expressions randomly within your time limit. Here is one possible approach:

For each random shuffle of 6 input numbers that we still have time for,
For each of the possible assignment of the operators
Try each of the order-of-operation groupings to create an expression.
If the expression equals a 3-digit number not already solved, store it as a solution.
At the end of 30 seconds, report how many expressions were tried and show the solutions.

k. While the program is running, print a period each time the program finds the solution to a new 3-digit number. This way, the user can observe the progress of the computation. Be sure to print a \n before reporting the final results.
l. On the class Web site, I'll provide you with a text file `arrangements.txt`, which indicates all of the possible ways that the order of operations can be specified using 2-6 input numbers. In this file, 'N' stands for number, and 'O' stands for operator. Place this input file in the same directory as the rest of the files for your program. Your program should open this file and use these arrangements as templates when creating expressions.
m. Optional: You may find it useful to write a second program that reads your output and verifies that the postfix expressions are valid.
n. When you are done with your program, please make an appointment to show me your work.
o. Have fun!

Example I/O (with excess periods and solutions removed):

```
java Driver 6 3 25 50 75 100
.....................................................................
.....................................................................
.....................................................................
.....................................

Number of expressions tried:  7995392
Number of solutions found:    812
100 = 50 75 6 + 100 25 - - * 3 /
101 = 50 75 + 6 100 * 25 / -
102 = 50 75 6 100 + + 25 - 3 / +
103 = 3 25 + 75 +

[hundreds of other solutions omitted here …]

997 = 25 3 - 6 50 100 + * 75 + +
999 = 6 100 25 / 3 50 * + * 75 +
```