Problem 143: Complex Code

Difficulty: Medium

Author: Matt Hussey, Ampthill, Reddings Wood, United Kingdom

Originally Published: Code Quest 2021

Problem Background

With a Code Quest problem, the main goal is to make sure your program gets the correct solution. It doesn't matter how clean your code is or (usually) how efficient it is; producing the correct output will get you points regardless.

In practice, however, software programs need to be maintainable. Writing neat code and creating detailed documentation helps others understand what your code does, which in turn makes it much easier for any developer to correct bugs in your code when they are found. Conversely, needlessly complicated code can be difficult to maintain, since it takes more time to figure out what the code is doing.

Problem Description

It's possible to analyze software code to determine its level of complexity. Two metrics used for this purpose are "cyclomatic complexity" and "nesting depth." Your team is working with systems engineers at Lockheed Martin to scan a section of code to determine if that code exceeds acceptable limits for these metrics.

Cyclomatic complexity is a measure of how many possible paths exist through a section of code. For example, consider the pseudocode below:

```
Print "Hello"
Print "World"
Print "This is not complex"
```

This code snippet contains three statements, but there's only one way to execute them; the program offers no alternative paths to take. It has a cyclomatic complexity of 1.

Now consider the snippet on the next page:

```
Print “I have a number”
If num > 2
{
      Print “It’s more than 2”
}
Else
{
      Print “It’s 2 or less”
}
```

In this snippet, the first print statement will always be executed. However, the program could execute either of the other two print statements, depending on the value of “num.” This creates two possible paths for the snippet to take, resulting in a cyclomatic complexity of 2.

Nesting depth indicates the maximum number of nested statements within a section of code. The first snippet above only included print statements; no nesting was involved. It has a nesting depth of 0. The second snippet included if and else statements. Each of those statements included a nested print statement. That snippet has a nesting depth of 1. The snippet below has a nesting depth of 3:

```
If num > 2
{
      If num > 3
      {
            If num > 4
            {
                  Print “It’s really big”
            }
            Else
            {
                  Print “It’s pretty big”
            }
      }
      Else
      {
            Print “It’s kinda big”
      }
}
Else
{
      Print “It’s not big”
}
```

The first two print statements are each nested within three layers of statements, resulting in the nesting depth of 3. Even though the code contains other nested statements, none are nested at a greater level than that, so they do not count towards the nesting depth. This snippet also has a cyclomatic complexity of 4, since there are four possible paths through the snippet; one if num is 2 or less, a second if num is 3, a third if num is 4, and a fourth if num is greater than 4.

Sample Input

The first line of your program's input, received from the standard input channel, will contain a positive integer representing the number of test cases. Each test case will include:

- A line containing three integer values separated by spaces:
  - L, a positive integer representing the number of lines of code to evaluate
  - C, a positive integer representing the highest acceptable cyclomatic complexity for the code snippet
  - N, a non-negative integer representing the highest acceptable nesting depth for the code snippet
- L lines containing a snippet of pseudocode, which will consist of a series of print, if, and else statements. Lines will not include indentation; the examples above were indented only to demonstrate the relevant concepts.
  - Print statements are a single line and will begin with the word "Print", which is then followed by a string of text wrapped in quotes (").
  - If statements cover multiple lines. The first line will start with the word "If", followed by a comparison. Comparisons will be made between a variable (containing only lowercase letters) and an integer number, and may use one of the operators >, <, >=, <=, ==, or !=. The next line will contain a left curly bracket ({). A later line will contain a matching right curly bracket (}), which terminates the if statement.
  - Else statements cover multiple lines, and will only occur immediately after the termination of an if statement. The first line will contain the word "Else". The next line will contain a left curly bracket ({). A later line will contain a matching right curly bracket (}), which terminates the else statement.

```
2
9 2 1
Print “I have a number”
If num > 2
{
Print “It’s more than 2”
}
Else
{
Print “It’s 2 or less”
}
11 2 2
If num > 3
{
If num > 4
{
Print “It’s really big”
}
Else
{
Print “It’s pretty big”
}
}
```

Sample Output

For each test case, your program must print a single line containing the following values, separated by spaces:

- An integer representing the snippet’s cyclomatic complexity
- An integer representing the snippet’s nesting depth
- If both of the above metrics were equal to or less than their limits (C and N, respectively), print the word “PASS”; otherwise, print “FAIL”

```
2 1 PASS
3 2 FAIL
```