# COSC 251 – Programming Languages
# Project 2
# Spring 2018

**Objective:** Use Python to solve a bevy of problems.

**Your Task:** The SMCM Programming Team once upon a time competed each fall in a programming competition hosted by colleges in our region. As part of their preparation, they solved a wide variety of problems all of which could be solved via Python without too many issues. For this project, you will provide solutions to 3 of these problems. You will be required to answer all three questions and each question is worth 33 1/3 points. As a general guideline, while the problems are all worth the same amount, their difficulty is not. Generally, the difficulty starts with Q1 as the easiest problem, and Q3 is the most difficult.

For all questions, input may be provided to your function through the parameter list, or through user input handled by your function. Pay attention to each description for information on which questions are which. Also, all output should be handled by your function, do not return any data. For all problems, you may not use any packages external to python.

**Q1:** If you've ever taken a chemistry course, you've seen this sort of thing denoting a chemical reaction:

$$2Na + 2H_2O = 2NaOH + H_2$$

The symbols 'H', 'O', and 'Na' stand for various kinds of elements; each term (separated by '+' and '=') represents a molecule; the subscripted integer numerals after each element (defaulting to 1) represent the number of that element in the molecule (though elements can occur multiple times – as in acetic acid, '$CH_3COOH$'); and the integer numeric coefficients in front of a molecule (again defaulting to 1) represent the number of participating molecules of the attached type. You are to write a program that checks such equations for balance. For example, your program will accept

$$6H_2O + 6CO_2 = 6O_2 + C_6H_{12}O_6$$

but indicate that

$$O_2 + H_2 = H_2O$$

is erroneous.

The input to your program will consist of equations of the form shown above, separated by whitespace, except that the equations themselves contain no whitespace and subscripted numerals are not written with subscripts. Each chemical element is denoted by a single upper-case letter, an upper-case letter followed by one letter, or an upper-case letter followed by two letters (Uuo for instance).

For each equation, produce a line of output that echoes the equation followed either by the phrase "balances" or "does not balance" in the format shown in the example below.

Example:

| Input: | Output: |
|---|---|
| 6H2O+6CO2=6O2+C6H12O6<br>2Na+2H2O=2NaOH+H2 C6H12O6=3C2H2+3O2 | 6H2O+6CO2=6O2+C6H12O6 balances<br>2Na+2H2O=2NaOH+H2 balances<br>C6H12O6=3C2H2+3O2 does not balance |

Method signature: Problem1(s)
No user input allowed

**Q2:** Consider the sequence of all words formed entirely of lower-case letters and having the following properties:

- A word *x* appears before a word *y* if *x* is shorter than *y*.
- Any two words of the same length appear in alphabetical order.
- The sequence contains exactly the words whose letters appear in strictly increasing order (for example 'a', 'ab', 'abc', but not 'ba' or 'bb').

To each word in this sequence, associate a positive integer index, starting with 1:

a → 1
b → 2
…
z → 26
ab → 27
ac → 28
…
az → 51
…
vwxyz → 83681

Your program is to read a series of lower-case words from one to five letters long, separated by whitespace. For each word read, if the word is invalid print the number 0, and otherwise print its index in the sequence.

Example:

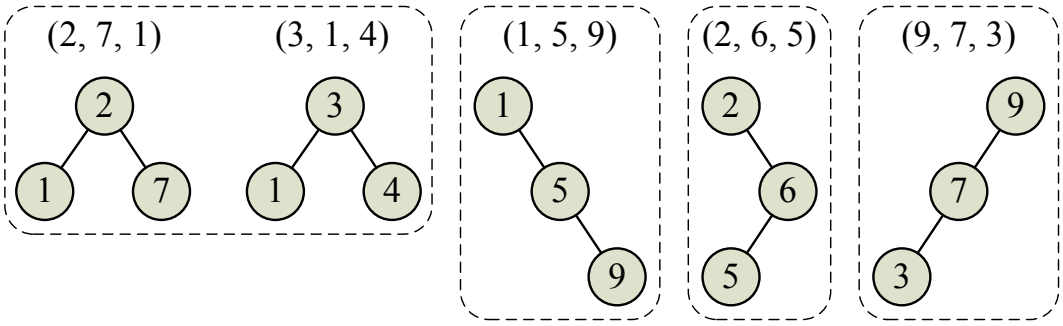
| Input: | Output: |
|---|---|
| z a<br><br>cat<br>vwxyz | 26<br>1<br>0<br>83681 |

Method signature: Problem2(s)
No user input allowed.

**Q3:** When looking at binary search trees (BST), it is sometimes helpful to determine (and group) trees that have the same topology. For this problem, you will take, as input, a series of numeric inputs that form one or more BSTs and output the total number of topologies that you generate. For instance, if I have five trees:

2 7 1
3 1 4
1 5 9
2 6 5
9 7 3

Then the resulting trees look like this:




And I can say that I have 4 total topologies.

The input is to be processed via keyboard, and the first line of the input contains two integers $n$ ($1 <= n <= 50$), which is the number of trees to analyze, and $k$ ($1 <= k <= 20$) which is the number of vertices in each tree. The next $n$ lines describe each tree. Each of those lines have $k$ distinct integers which are the values of each of the vertices.

Examples:

| Input: | Output: |
|---|---|
| 5 3<br>2 7 1<br>3 1 4<br>1 5 9<br>2 6 5<br>9 7 3 | 4 |
| 3 4<br>3 1 2 40000<br>3 4 2 1<br>33 42 17 23 | 2 |

Method signature: Problem3()

**Deliverables:** your Python source. All three sets of code should be stored in a single Jupyter notebook named Proj2.ipynb, following the above method signatures.

**Expectations:** The code should be clean, concise, well-commented and correct. If you use an outside source, be sure to document that source. Significant use of outside sources will result in a deduction. Grading rubric will be provided a week ahead of the due date. A driver with the input from the examples will be provided shortly. You are allowed to work in teams of up to three for this project. If you choose to work in a team, one member of the team is required to email me with who they are working with by 5pm, 3/1.

**Learning Targets:** Python development experience, classic problem solving, and a ton of reading comprehension.

**Credit:** Collegiate programming competitions.

**DUE: March 8th, 11:59pm via Blackboard, team information due 5pm 3/1 via email.**