COSC 251 – Programming Languages
Project 2
Spring 2019

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 will be provided to your function through the parameter list Also, all output should be handled by your function, do not return any data. You must match the output format exactly as it is shown in the examples or risk a deduction. For all problems, you may not use any packages external to python. You do not need to worry about errors in the input strings.

As a note – your professors and TAs will not walk you through how to solve these problems! If there are specific questions in regards to how the examples fit the question descriptions, those are fair game. We will also help debug syntax issues or confusion on how certain python functions work, but we will not help with logic errors.

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 2Na+2H2O=2NaOH+H2 C6H12O6=3C2H2+3O2	6H2O+6CO2=6O2+C6H12O6 balances 2Na+2H2O=2NaOH+H2 balances C6H12O6=3C2H2+3O2 does not balance

Method signature: Problem1(s)
No user input allowed

Q2: Recent recessions have not been kind to entertainment venues, including the gambling industry. Competition is fierce among casinos to attract players with lots of money, and some have begun to offer especially sweet deals. One casino is offering the following: you can gamble as much as you want at the casino. After you are finished, if you are down by any amount from when you started, the casino will refund ***x%*** of your losses to you. Obviously, if you are ahead, you can keep all your winnings. There is no time limit or money limit on this offer, but you can redeem it only once.

For simplicity, assume all bets cost 1 dollar and pay out 2 dollars. Now suppose ***x*** is 20 (this will not always be true, of course). If you make 10 bets in total before quitting and only 3 of them pay out, your total loss will be 3.2 dollars. If 6 of them pay out, you have gained 2 dollars.

Given ***x*** and the percentage probability ***p*** of winning any individual bet, write a program to determine the maximum expected profit you can make from betting at this casino, using any gambling strategy.

The input is a single string sent to your method. A string consists of the refund percentage ***x*** ($0 <= x < 100$) followed by the winning probability percentage ***p*** ($0 <= p < 50$). Both ***x*** and ***p*** have at most two digits after the decimal point. The output will be the maximum expected profit with an absolute error of at most 10^{-3}.

Example:

Input:	Output:
0 49.9 50 49.85	0.0 7.10178453

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

Q3: The Industrial Computer Processor Company offers very fast, special purpose processing units tailored to customer needs. Processors of the ***a***-C-***m*** family (such as the 1-C-2, and the 5-C-3) have an instruction set with only two different operations:

A add ***a***
M multiply by ***m***

The processor receives an integer, executes a sequence of A and M operations (the program) that modifies the input, and outputs the result. For example, the 1-C-2 processor executing the program AAAM with the input 2 yields the output 10 (the computation is 2 -> 3 -> 4 -> 5 -> 10), while the 5-C-3 processor yields 51 with the same program and input (2 -> 7 -> 12 -> 17 -> 51).

You are an ***a***-C-***m*** programmer assigned to a top secret project. This means that you have not been told the precise computation your program should perform. But you are given particular values ***p, q, r,*** and ***s*** and the following conditions:

1. The input is guaranteed to be a number between ***p*** and ***q***.
2. The output must be some number between ***r*** and ***s***.

Given an ***a***-C-***m*** processor and the numbers ***p, q, r,*** and ***s,*** your job is to construct the shortest ***a***-C-***m*** program which, for every possible input *x* such that $p <= x <= q$, yields some output *y* such that $r <= y <= s$.

If there is more than one program of minimum length, choose the one that comes first lexicographically, treating each program as a string of As and Ms.

The input will be a single string containing six integers as noted above (***a m p q r s***) from 0 to 10000000000 with ***p*** <= ***q***, and ***r*** <= ***s***. The output will be the smallest A&M program. If the best program uses no operations, print "empty", and if there is no program meeting the specifications, print "impossible". Format your solution string alternating between "nA" and "nM" where n > 0. This indicates n consecutive A or M operations before switching.

Example:

Input:	**Output:**
1 2 2 3 10 20	1A 2M
1 3 2 3 22 33	1M 2A 1M
3 2 2 3 4 5	impossible
5 3 2 3 2 3	empty

Method signature: Problem3(s)
No user input allowed

Deliverables: your Python source. All three sets of code should be stored in a single python file named Proj2.py, 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. A driver with the input from the examples will be provided shortly. **YOUR CODE MUST WORK WITH OUR DRIVER.** As before, you should run your code on Prometheus to verify that everything is working ok for grading purposes. 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 their professor with who they are working with by 5pm, 2/27.

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

Credit: Collegiate programming competitions.

DUE: March 6th, 11:59pm via Blackboard, team information due 5pm 2/27 via email.

Potential Deductions:
-30 each problem that does not interpret
-25 each problem that crashes
-20 lack of comments
-15 Any problem that requires an inordinate amount of execution time on Prometheus (> 10 minutes)
-75 Does not work with the driver
-20 each problem with incorrect output (outputs must match example format exactly)
-15 each problem that does not parse input correctly