COSC 251 – Programming Languages
Project 2
Spring 2017

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.

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. Some problems also have a "code golf" aspect to them, you must get your solutions under a particular character count, or you will be penalized. For all problems, you may not use any packages external to python.

Q1: My birthday is coming up. Alas, I am getting old and would like to feel young again. Fortunately, I have come up with an excellent way of feeling younger: if I write my age as a number in an appropriately chosen base *b*, then it appears to be smaller. For instance, suppose my age in base 10 is 32. Written in base 16, it is only 20!

However, I cannot choose an arbitrary base when doing this. If my age written in base *b* contains digits other than 0 to 9, then it will be obvious that I am cheating, which defeats the purpose. In addition, if my age written in base *b* is too small, then it would again be obvious that I am cheating.

Given my age *y* and a lower bound *l* on how small I want my age to appear, find the largest base *b* such that *y* written in base *b* contains only decimal digits, and is at least *l* when interpreted as a number in base 10.

The input consists of a single string passed to the function containing two base 10 integers *y* ($10 <= y <= 10^{18}$) and *l* ($10 <= l <= y$). Print the largest base *b* as described above.

Examples:

Input:	Output:
32 20	16
2016 100	42

Method signature: Problem1(s)
No user input allowed
Par: 318, including whitespace

Q2: During the October Revolution of 1917, the Bolsheviks used a simple transposition cipher to send messages. The message was broken into 25-character blocks. The end of the message would have additional characters added until the message length was a multiple of 25. Each block was then handled independently.

The message was written in a 5x5 grid. Spaces between words were ignored.
Message: `THE COMMISSAR SAYS HELLO`

Written in grid, with padding characters added:

```
THECO
MMISS
ARSAY
SHELL
OABCD
```

The encryption key is a permutation of the integers 1-5, which determines the order in which the *columns* are read to produce the ciphertext:

Key: `54123`
`OSYLD CSALC TMASO HMRHA EISEB`

You are to take in a string with a Bolshevik-encrypted message. Each string is written with the key (a permutation of the integers from 1-5), followed by the 25-character message in 5-character groups. The message will be entirely alphabetic (no digits or punctuation), all upper-case. Our sample message would be presented as

`54123 OSYLD CSALC TMASO HMRHA EISEB`

Your output is the decrypted message, also in 5-letter groups, each group separated by a single space with a newline (with no trailing space) at the end of each message. You do not need to strip out the padding.

`THECO MMISS ARSAY SHELL OABCD`

Examples:

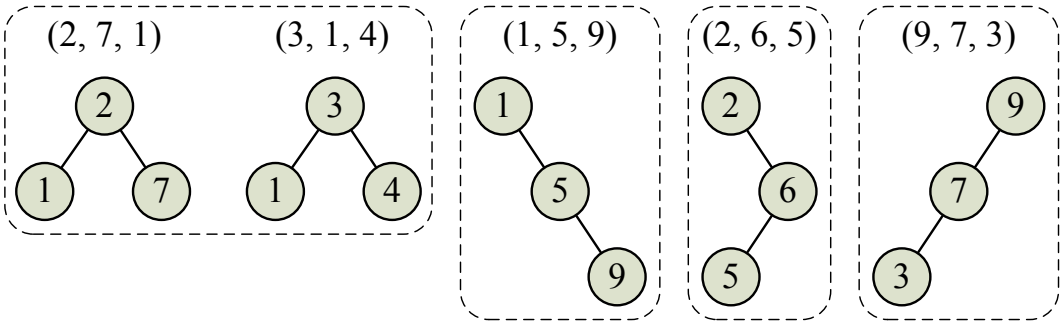
Input:	Output:
54123 OSYLD CSALC TMASO HMRHA EISEB	THECO MMISS ARSAY SHELL OABCD
41532 IEVEA AATST SAENA GRNMA IGITA	AIGIS AGREA TINVE STMEN TAAAA

Method signature: Problem2(s)
No user input allowed.
Par: 166 characters, including whitespace.

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 2 7 1 3 1 4 1 5 9 2 6 5 9 7 3	4
3 4 3 1 2 40000 3 4 2 1 33 42 17 23	2

Method signature: Problem3()
Par: No maximum character requirement

Deliverables: your Python source. All three sets of code should be stored in a single 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. 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, 2/27.

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

Credit: Collegiate programming competitions.

DUE: March 10th, 11:59pm via Blackboard, team information due 5pm 2/27 via email.