COSC 251 – Programming Languages
Project 2
Spring 2013

Objective: Use Python to solve a bevy of problems.

Your Task: The SMCM Programming Team competes each fall in a programming competition hosted by colleges in our region (talk to Lindsay if you're interested in joining the team). As part of their preparation, they solve 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 4 of these problems. You will be required to answer all four questions and each question is worth 25 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.

Q1: Suppose you have a network of friends who are all friends with each other on some social network. The friends all, of course, share memes such as kittens, rage comics and George Takei reposts on their walls which are seen by all in the network. Each friend decides whether or not to repost a meme that they see on someone's wall based on how much they have already seen the meme as well as the awesomeness of the meme. Each friend has a popularity threshold that must be reached before they will repost the meme to their wall. Each meme is started by an 'invisible' poster, whom all friends on the network are friends with but who is not represented on the network.

However, after a certain point, the memes get stale and no one will repost them anymore. That's why when all the friends decide if they will repost the following formula.

Friendi will repost if friendi's threshold $t < a(10n - n^2)$, where a is the awesomeness coefficient of the meme and n is the total number of times that the meme has been posted in the network. The count starts at 1, as it includes the invisible poster. So, if friend5 has threshold 10, and meme3 has awesomeness coefficient 0.5 and has been posted 3 times, then $a(10n - n^2) = 0.5(30 - 9) = 10.5$. So friend5 will repost this meme. Once a meme has been posted, that friend will not post the meme again. Note: the friends all consider the meme at the same exact moment, and all post (or not post) at the same moment. It may help to consider the posting in rounds rather than each individual friend considering each individual meme.

The popularity measure of a meme is taken after no one will repost the meme anymore. It is simply the number of friends who posted the meme to their wall divided by the total number of friends. That is, if there are m friends and p total postings, then pop = p/m. We say the meme was popular if pop > 0.5. The initial 'invisible poster' does not count towards the popularity of the meme, but it does count when computing n for the reposting formula above.

Task: For a given set of m friends with specified posting thresholds (where $0 < m <= 25$), take in q awesomeness quotients ($0 < q <= 10$, quotients run from 0 to 1, inclusive). The first input will be the number m of friends, followed by a line with the posting thresholds of each friend separated by spaces. The next line is the number q of awesomeness thresholds followed by a line of awesomeness thresholds separated by spaces. Print out whether the given meme was popular on this network along with the popularity of that meme in the format "[no/yes]: [popularity]".

Example:

Input1:	Output1:
10 3 25 1 12 16 17 5 0 21 4 3 0.22 1 0.66	no: 0.5 yes: 0.9 yes: 0.6
Input2:	Output2:
1 12 3 0.22 1 0.66	no: 0.0 no: 0.0 no: 0.0

Method signature: Problem1()
User input required.

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 cat vwxyz	26 1 0 83681

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

Q3: You are a lone human, without guns or even a crowbar, who is attempting to run across a field of zombies and not get eaten. The field is represented as a square 2D grid, where each grid cell contains a number representing the number of zombies on that square. You start in the lower-left hand corner, and need to reach the upper-right hand corner.

For each square you need to cross on the path that has a non-zero number of zombies, your chance of survival will be reduced by 5% per zombie. So if you cross two squares with one zombie on them or one square with two zombies on it, your overall chances of survival will be 90%. The starting and ending squares will never have zombies on them.

Movement can only consist of moving up or right, with no diagonal moves allowed. Thus each possible path will have exactly 2(k-1) moves, where k is the width and height of the grid.

The input will consist of a number, k, which is the vertical and horizontal size of the grid where 0 < k <= 15. This will be followed by a newline character and k lines of k numbers between 0 and 2 (inclusive), separated by spaces.

Output should print a single number between 0 and 100, representing the probability of survival for the best possible path.

Example:

Input1: 3 0 0 0 0 1 0 0 2 0 Input2: 4 1 0 2 0 0 2 1 0 0 1 0 0 0 1 0 1	Output1: 100 Output2: 95

Method signature: Problem3()
User input required.

Q4: A Silly Array Sort - You are to create a function to take in a user-generated list of integers, which you will then perform a silly sort on. The array must be even in length and contain equal numbers of odd and even numbers.

You should prompt the user for the size of the array, then for its contents in no particular order. You may not assume that a valid array size and valid contents will be entered. You should check for errors with array size, array contents, and non-integer input from the user.

Your output should be all of the numbers that you were given, with alternating odds and evens. The output should start with an even number and end with an odd number. Within this, the evens should be sorted in ascending order, while the odds should be sorted in descending order. You should present the output as an array (between brackets with individual numbers separated by commas).

Input:	Output:
6 6 3 4 2 5 1	[2, 5, 4, 3, 6, 1]

Method signature: Problem4()
User input required.

Deliverables: your Python source. All four sets of code should be stored in a single file named Proj2.py, following the above method signatures. Do not provide any prompts for input and follow input specifications precisely.

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 pairs for this project. If you choose to work with someone, one member of the pair should email me that information by 5:00pm, February 22nd.

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

Credit: 2011-2012 Programming Team, Lindsay Jamieson and the 2010 UC-Berkeley Programming Competition. Some problems are from a UVa repository of problems.

DUE: March 7th, 11:59pm via Blackboard