Q1: Sum of Numbers [2 marks]

The input data is represented as a 2D list of integers as follows: 2 1 3 4 9 8 6 2 7 The matrix above is represented like this in Python: m = [[2, 1, 3], [4, 9, 8], [6, 2, 7]]

You need to produce an output 2D list of integers, with each cell in the output corresponding to the cell at the same column and row in the input, as explained below.

For each valid cell: (a) Identify the numbers to the right (across). (b) Identify the numbers below (down). (c) Compute the sum of all these identified numbers

e.g. 1: for the cell in the first row and first column (2), the sum for 2 across is 2 + 1 + 3 = 6. The sum for 2 down is 2 + 4 + 6 = 12. Add across (6) and down (12) and store the value 18 in the corresponding cell in the output.

e.g. 2: for the cell in the last row and first column (6), the sum for 6 across is 6 + 2 + 7 = 15. The sum for 6 down is 6, because there is no further number. Add across (15) and down (6) and store the value 21 in the corresponding cell in the output.

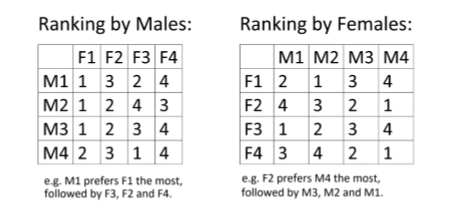
The function should return the following output 2D list. output = [[18, 16, 21], [31, 28, 23], [21, 11, 14]]

Requirements: Edit the function q1\_recursive(m, output, row, col) in p1q1.py that takes in a 2D list (m), and returns the corresponding output 2D list. You can use a combination of recursion and iteration to solve the problem but recursion must be used as part of the solution. Note: do not edit the q1(m) function in p1q1.py.

Q2: The Stable Marriage Problem (Again) [2 marks]

During week 1’s lecture, you were briefly introduced to the stable marriage problem as well as how it can be applied to practical situations such as kidney-matching. This is a well-documented problem, and you are required to find out more about the stable marriage problem from online sources yourself.

In this context, n males and n females rank each member of the opposite sex from 1 (most preferred) to n (least preferred). For example, the following tables show the preferences of 4 couples (n=4).



The ranked preferences are represented in the form of a 2D list. The following shows how the two tables above are represented: [[m1, 1, 3, 2, 4], [m2, 1, 2, 4, 3], [m3, 1, 2, 3, 4], [m4, 2, 3, 1, 4], [f1, 2, 1, 3, 4], [f2, 4, 3, 2, 1], [f3, 1, 2, 3, 4], [f4, 3, 4, 2, 1]]

For these preferences, solution (a) below is unstable, and solution (b) is stable: (a) (M1-F1), (M2-F3), (M3-F2), (M4-F4) – unstable solution (b) (M1-F3), (M2-F1), (M3-F4), (M4-F2) – stable solution

Proposed solutions to the problem can also be represented as 2D lists. The two solutions above are represented like this: Solution (a): [[m1, f1], [m2, f3], [m3, f2], [m4, f4]] Solution (b): [[m1, f3], [m2, f1], [m3, f4], [m4, f2]]

Assumptions:  The preferences are always valid. i.e. there are no “ties”, and each female is only ranked once by each male (and vice versa). You will not see invalid preferences such as [[m1, 1, 1, 2, 4],…] or [[m1, 1, 3, 5, 4],…] or [[m1, 0, 1, 2, 3],…]  The preferences may be in any order. E.g. the preferences of f3 may appear before the preferences of f2. So this is possible: [[f3, 1, 2, 3, 4], [m1, 1, 3, 2, 4], [m3, 1, 2, 3, 4],…]  In the solution list, the males will always come before the female partner. You will not see this: [[f1, m1],…]. However, the pairs may be in any order. So this is possible: [[m2, f1], [m1, f3], [m3, f4], [m4, f2]]  In the solution, each member will only appear once, and males will always be paired with females.

For this question, you are not required to come up with a correct solution to the problem (in fact, existing solutions such as the Gale-Shapley algorithm are well known and well documented). Instead, you are required to verify if a particular proposed solution to the stable marriage problem is correct. A correct solution is one in which all pairs in the solution are stable.

Requirements: Edit the function is\_stable in p1q2.py. is\_stable takes in n, the preferences, and a proposed solution to the problem. It is supposed to return True (if the solution is stable) or False (if not).

e.g. >>> pref = [[m1, 1, 3, 2, 4], [m2, 1, 2, 4, 3], [m3, 1, 2, 3, 4], [m4, 2, 3, 1, 4], [f1, 2, 1, 3, 4], [f2, 4, 3, 2, 1], [f3, 1, 2, 3, 4], [f4, 3, 4, 2, 1]] >>> solution\_a = [[m1, f1], [m2, f3], [m3, f2], [m4, f4]] >>> solution\_b = [[m1, f3], [m2, f1], [m3, f4], [m4, f2]] >>> is\_stable(4, pref, solution\_a) False >>> is\_stable(4, pref, solution\_b) True

Q3: Lab 4 Extended! [5 marks]

This is an extension of lab 4, so you must be familiar with the question in lab 4. For this question, each Twitter user is given two additional attributes: a cost (c) and value (v). c is how much a user is asking to be paid in order to tweet your advertisement, and v is the approximated spending ability of the user based on his profile and income. We prefer users with high v to receive your advertisement.

Instead of selecting 5 users to tweet your advertisement, you are now given a budget to pay your advertisers. You can select any number of users to be your advertisers as long as their total c does not exceed the budget. Also, the objective of your marketing campaign has changed: in lab 4, you wanted to maximize the number of users who will get the message. For this question, you want to maximize the sum of v for all the users who will get your message.

e.g. 1: (n=11) Budget allocated = 10. User ID User IDs of Direct Followers Cost Value 0 2 1 9 1 0, 3 2 4 2 0, 1 1 7 3 1, 2, 4, 5 2 5 4 1, 6, 10 2 6 5 - 1 2 6 7, 8 7 5 7 - 2 2 8 - 2 2 9 8 5 2 10 9 3 4

e.g. 1(a): This is an “incorrect” solution: [3, 4, 6, 10]. Total cost of these 4 users = c3 + c4 + c6 + c10 = 2 + 2 + 7 + 3 = 14, which is > the allocated budget of 10. Solutions which bust the allocated budget are considered incorrect.

e.g. 1(b): This is a “correct” solution: [0, 1, 2, 3, 4]. Total cost of these 5 users = c0 + c1 + c2 + c3 + c4 = 1 + 2 + 1 + 2 + 2 = 8, which is <= the allocated budget of 10. The additional users who will get the message are users 5, 6 and 10, making it a total of 5+3, or 8 users who will get the message. Remember that for this question, the actual number of users who get the message is irrelevant. We are interested in the total value of the users who get the message. For this case, total value = = (v0 + v1 + v2 + v3 + v4) + (v5 + v6 + v10) = 9 + 4 + 7 + 5 + 6 + 2 + 5 + 4 = 42 The quality score for your solution will be the total value, and you strive to maximize the quality score.

Requirements: Edit the function select\_advertisers in p1q3.py. select\_ advertisers takes in budget, followers (as for lab 4), costs (a list of costs for each user), and values (a list of values for each user). It is supposed to return a list of user IDs selected by your algorithm.

e.g. >>> followers = [[2], [0,3], [0,1], [1,2,4,5], [1,6,10], [], [7,8], [], [], [8], [9]]

>>> c = [1, 2, 1, 2, 2, 1, 7, 2, 2, 5, 3] >>> v = [9, 4, 7, 5, 6, 2, 5, 2, 2, 2, 4]

>>> s = select\_advertisers(10, followers, c, v) # budget is 10

There are other functions used in p1q3\_main.py that may be helpful. Check them out yourself.

Requirements for Written Report:  You are required to submit ONE written report (in PDF format only) for the whole assignment to eLearn by the same deadline. Name your report <Your name>.pdf.  Explain your algorithms adopted to approach Q2 and Q3. If possible, compute the time complexity of your algorithm and show how this value is derived. Besides your complexity calculation, you will be given credit for clarity of explanation, novelty of approach and evidence of analytical thinking. Use diagrams and examples where relevant. You are not required to explain Q1 in your report.