ITI1120 Winter 2018 - ASSIGNMENT 4

Read the instructions below carefully. The instructions must be followed. This assignment is worth **5%** of your grade. The assignment is due on `Mar 14th, 8:00am`. *No late assignments will be accepted.*

The goal of this assignment is to learn and practice (via programming) the concepts that we have learned thus far. For this assignment that includes: 2D lists, file processing, strings, building somewhat bigger programs, finding algorithms/solutions for fundamental problems like searching, frequency counting …

As part of this assignment you find two files: NYT-bestsellers.txt and `part2-example-run.py`
 Their purpose will be explained below.

This assignment has two parts. In part 1 you will implement one function working on a 2D list (matrix). In part 2, you will implement a bigger program. Put the below two required documents into a folder called `a4_xxxxxx`, zip that folder and submit it as explained in lab 1. (In particular, the folder (and thus your submission) should have the following files:
`a4_ap4_xxxxxx.py`, (for part 1)
`a4_NYT_xxxxxx.py` (for part 2)

For each function that you design for this assignment you have to include docstrings that specify:
- type contract
- description about what the function does (while mentioning parameter names)
- preconditions, if any.

For this assignment, you do not need to submit a text file as evidence that you tested your functions. But as always, you are strongly encourage to test all your functions in Python shell as you have learnt in the previous assignments.

As always, your programs must run without syntax errors.

****
PART 1 (15 points): PATTERN RECOGNITION — arithmetic progression of length at least 4
****

=====================
Question 1: (10 points)
=====================

Implement a Python function called `ap4` that takes a 2D list `m` of numbers as input (parameter) representing a matrix and tests if `m` has at least four consecutive numbers that form an arithmetic progression either horizontally, vertically or diagonally (in either of the two diagonal directions \ or /). If yes, the function returns a 2D list containing 4 locations in the matrix where one such sequence is found. This return 2D list has to be sorted by using python's sorted function (so that your function can be automatically tested). If no sequence is found, the function returns empty 1D list. You may assume that `m` represents a matrix, meaning m will not have rows with different number of elements.
https://en.wikipedia.org/wiki/Arithmetic_progression

Place your function `ap4` in one file called `a3_ap4_xxxxxx.py`.

Examples of function calls:
```
>>> ap4([[0,  10, 1,  1, 0],
         [1,   7, 2,  2, 1],
         [4,   4, 3,  5, 2],
         [9,   1, 4, 10, 3],
         [16, -2, 5, 17, 4],
         [25, -5, 6, 26, 5]])
[[0, 1], [1, 1], [2, 1], [3, 1]]

>>> ap4([[2, 5, 2, 0, 0],
         [5, 0, 4, 4, 4],
         [2, 1, 3, 3, 3],
         [1, 5, 0, 4, 2],
         [4, 2, 1, 5, 5]])
[[0, 1], [1, 2], [2, 3], [3, 4]]
```

```
>>> ap4([[2, 5, 2, 0, 0, 3],
         [5, 0, 4, 4, 0, 0],
         [2, 1, 3, 0, 3, 7],
         [1, 5, 0, 4, 2, 8],
         [4, 0, 1, 5, 5, 2]])
[[1, 4], [2, 3], [3, 2], [4, 1]]

>>> ap4([[1, 2],
         [3, 4]])
[]

>>> ap4([[-48, -63, -80, -99, -120],
         [-41, -56, -73, -92, -113],
         [-22, -37, -54, -73, -94],
         [15,    0, -17, -36, -57],
         [76,   61,  44,  25,   4],
         [167, 152, 135, 116, 95]])
[]

>>> ap4([[-19, -8, 35, 35, -41, 19, 30, -43, 25],
         [-12, -47, 14, -6, 31, 16, -40, 0, -38],
         [40, 38, 26, -13, 47, -13, 40, 25, -26],
         [37, -21, -40, 43, -7, -28, -33, -3, 50],
         [10, -37, 37, -11, -40, -14, 5, 42, -43],
         [49, -34, 27, 31, 25, -31, 36, -48, -5],
         [23, -16, -47, 30, 46, 13, -30, 0, 23]])
[]
```

****

PART 2 (90 points): Processing New York Times bestseller books list

****

The New York Times newspaper (NYT) has published bestseller lists since October 12, 1931:
https://en.wikipedia.org/wiki/The_New_York_Times_Best_Seller_list

You will design, implement and test a program which allows the user to search a subset of the books which have appeared in the New York Times best seller lists and well as look for authors with most best sellers. All of you program must be in one file called `a3_NYT_xxxxxx.py` Unlike the previous assignment there is no starting code provided.

The data set that you will need to process is contained in the file called NYT-bestsellers.txt
While the file has data from 1942 to 2013, and thus misses some recent years, you may assume that the list is complete. (You program will be tested with this list and a similar list which has the same format)

Each line of NYT-bestsellers.txt contains the information for a separate book, which includes: title, author, publisher, date it first reached #1 on one of the best seller lists, and category (fiction or nonfiction). There is a tab character between fields.

The program will input (read) the data set from the file NYT-bestsellers.txt and construct a list of books (as a 2D list).

After constructing the 2D list (list of list) of books, the program will display a menu of options. The menu options are:

`1: Look up year range`

Prompt the user for two years (a starting year and an ending year), then display all books which reached the #1 spot between those two years (inclusive). For example, if the user entered “1970” and “1973”, display all books which reached #1 in 1970, 1971, 1972 or 1973.

`2: Look up month/year`

Prompt the user to enter a month and year, then display all books which reached #1 during that month. For example, if the user entered “7” and “1985”, display all books which reached #1 during the month of July in 1985.

`3: Search for author`

Prompt the user for a string, then display all books whose author's name contains that string (regardless of case). For example, if the user enters "ST", display all books whose author's name contains (or matches) the string "ST", "St", "sT" or "st".

`4: Search for title`

Prompt the user for a string, then display all books whose title contains that string (regardless of case). For example, if the user enters "secret", three books are found: "The Secret of Santa Vittoria" by Robert Crichton, "The Secret Pilgrim" by John le Carré, and "Harry Potter and the Chamber of Secrets".

`5: Number of authors with at least x bestsellers`

Prompt the user for a positive integer `x`, then display authors who have at least `x` bestsellers.

`6: List y authors with the most bestsellers`

Prompt the user for a positive integer `y`, then display the first `y` authors in terms of the highest number of bestsellers. If the `(y+1)` st author, in terms of the highest number of bestsellers, has the same number of bestsellers as `y` th author, you still only display y authors only and not `y+1` or more authors.

`Q: Quit`

Part 2 Requirements and more details:

1. Your program will consist of at least seven functions: a separate function to process each of the six menu options listed above and a helper function for option 5 and 6 (to be described below). One of the input parameters for functions solving options 1 - 4 has to be the 2D list of books.

2. You may lists and tuples in your program, but you may not use other collections (such as a dictionaries or maps).

3. Be sure to display the books in a reasonable and readable manner. Since the data file is "real data" you will find stray spaces such as at the beginning or end of some strings (e.g. author name) — the `strip()` method is handy for cleaning that up.

4. If no books are found for a particular search, your program will display an appropriate message (rather than simply displaying nothing).

5. Your program will continue to execute until the user selects "Q" (or "q") as the menu option.

6. Be sure to prompt the user for the inputs in the specified order. Also, your program cannot prompt the user for any other inputs.

7. Your program will handle erroneous user inputs. If there are any problems with a particular user input, your program will display an appropriate message and repeat the last question.

8. For 5 and 6, you may assume that the names of all authors are correct and spelled the same way. For example, if there is an enter with a book written by Kurt Vonnegut Jr. and another entry with a book written by Kurt Vonnegut, you may assume these are two distinct authors since their names are not spelled the same way.

9. To solve 5 and 6 you must have a helper function whose definition is like this:

```
def frequency(books):
    '''(2D_list)->2D_list'''
    f=[]
    #YOUR CODE GOES HERE
    return f
```

This functions `frequency` must take as input the 2D list of books and return a 2D list called `f` where each element of `f` is list of length two, with one element the name of an author and another the number of bestsellers that authors has. The number of elements in `f` must be the same as the number of distinct authors in NYT-bestsellers.txt

Then functions solving options 5 and 6, must not use `books` list. Instead they must user this list `f`, created in the helper function `frequency` as the means of solving the problems specified in options 5 an 6.

10. No global variables are allowed in functions: only use variables that are in the function's namespace. That is, only use variables that are parameters or are otherwise added to the function's namespace such as through assignment or iteration

11. The program must meet all the requirements implied by the run in `part2-example-run.py`

==
EXAMPLE RUNS OF YOUR PROGRAM:
==

For an example run see the file that comes with this assignment called
`part2-example-run.py`

As specified above (in 11) your program must meet all the requirements as implied by the run above run.