CSCI 1133, Spring 2023
Homework 06
Due: 11:55 pm, Tuesday February 28, 2023

Due Date: Submit your solutions to Gradescope by 11:55 pm, Tuesday, February 28. Do not upload anything to Canvas and PLEASE be sure to use proper naming conventions for the file, classes, and functions. We will NOT change anything to run it using our scripts.

Late Policy: Submissions will be accepted up to 24 hours late (11:55 pm on Wednesday) for a 1 point penalty, up to 48 hours late (11:55 pm on Thursday) for a 2 point penalty, or up to 72 hours late (11:55 pm on Friday) for a 3 point penalty. Submissions past 11:55 pm on Friday will not be accepted.

Unlike the computer lab exercises, this is not a collaborative assignment. See the syllabus and read section “Academic Dishonesty” for information concerning cheating. Always feel free to ask the instructor or the TAs if you are unsure of something. They will be more than glad to answer any questions that you have. We want you to be successful and learn so give us the chance to help you.

Purpose: The primary purpose of this assignment is to get practice with python looping constructs Iterating over both sequences and ranges of numbers.

Instructions: This assignment consists of 3 problems, worth a total of 30 points. All documentation and code must be put in a file named hw06.py and submitted on Gradescope.

Because your homework file is submitted and tested electronically, the following are very important:

- Submit the correctly named file through Gradescope by the due date deadline.
- Your program should run without errors when we execute it using Python 3.

Submit your work to Gradescope under the Homework 06 assignment. Make sure that your file is named correctly: if you don’t match the file name exactly then the testing script won’t work and you’ll lose points.

The following will result in a score reduction equal to a percentage of the total possible points:

- Incorrectly named/submitted source file (up to 20%)
- Failure to execute due to syntax errors (up to 30%)
- Bad coding style (missing documentation, meaningless variable names, etc.) (up to 30%)

A note on break: While the break keyword can be a useful tool for getting out of loops early, it tends to be misused by intro students to get out of writing a proper loop condition. So for the purposes of this class, the break keyword is considered bad style and will lose you points.

Documentation:
Use the following template for EVERY function that you write. The docstring below should be placed inside of the function, just after the function signature.

```
'''
Purpose:
Parameter(s):
Return Value:
'''
```

Problem A. (10 *points*) **List Difference**

Write a function `list_difference(numlist1, numlist2)` that takes in two lists as parameters. These lists contain numeric values and have the same length. Have the function return a new list that contains the difference between respective items in `numlist1` and `numlist2`, without mutating either of the given lists.

You are welcome to add an `if __name__ == '__main__'` block and test cases for this and all functions in this assignment but this is not required.

`Examples (text in bold is returned):`

```
>>> alist = [5,6,7]
>>> blist = [2,2,2]
>>> list_difference(alist, blist)
[3, 4, 5]
>>> alist
[5, 6, 7]
>>> blist
[2, 2, 2]
>>> list_difference([0,1,1,2], [3,5,8,13])
[-3, -4, -7, -11]
>>> list_difference([],[])
[]
```

Problem B. (10 *points*) **Searching a List**

Write a function `larger_decrement(numlist, n)` that takes two parameters, a list of integers `numlist`, and an integer `n`.

If the given list contains any number larger than `n`, then have the function mutate the list by decreasing the largest item in the list by 1. If the list has two or more elements tied for the largest, only decrease the first occurrence. Then the function should return `True`.

If the given list does not contain any number larger than n, it should return `False`.

Hints:

- The autograder will be expecting you to return a boolean value, make sure that you use the specific values `True` and `False`

Hints:

- You are not permitted to use built-in functions like max() or sorted() that trivialize the task of finding the largest element in the list.

You are welcome to add an `if __name__ == '__main__'` block and test cases for this and all functions in this assignment but this is not required.

Examples (text in **bold** is returned):

```
>>> alist = [5, 20, 74, 81, 0, 81, 3]
>>> larger_decrement(alist, 50)
True
>>> alist
[5, 20, 74, 80, 0, 81, 3]
>>> blist = [1, 3, 5]
>>> larger_decrement(blist, 6)
False
>>> blist
[1, 3, 5]
```

Problem C. (10 *points*) **Word Mixing**

Write a function `word_mix(wordlist1, wordlist2)` that takes as parameters two lists that contain strings. Have the function return a single string that is the result of concatenating alternating strings from each list, starting with `wordlist1`, inserting a space character between strings. If the lists do not contain the same number of strings, then continue concatenating strings from the list with more items. Make sure that your final string does not have an extra space character at the end.

Hints:

- You may have to have multiple loops to handle the case where the lists have unequal lengths.
- There are several ways to deal with an extra space at the end of the string. These include an if statement, slicing, or certain string methods for those of you reading ahead.

You are welcome to add an `if __name__ == '__main__'` block and test cases for this and all functions in this assignment but this is not required.

```
Examples (text in bold is returned):
>>> word_mix(['the','brown'],['quick','fox'])
'the quick brown fox'
>>> word_mix(['the','brown','jumped','over'], ['quick','fox'])
'the quick brown fox jumped over'
>>> word_mix(['the','brown'], ['quick','fox','jumped','over',
'the','lazy','dogs'])
'the quick brown fox jumped over the lazy dogs'
```