Lab/Homework 3

# **Deadline**: 23:59 pm, Sunday, Nov. 19.

# **What to submit:**

A report with answers to each exercise and corresponding python

program(.py file), packaged into a zip file. Named zip as

“class\_name\_HW3” (for example:AI1\_刘晔\_HW3), submit to TA (Jiashuo Zheng).

# **Requirements on Coding:**

1. Adding header to each .py file.

"""

xxxx.py

author:

date:

description:

"""

1. Please add a space around the operator and after the comma.
2. Add a blank line between code of different functions
3. Indent your code blocks with 4 spaces. Never use tabs or mix tabs and spaces.

# **Exercise 3.0 RotateString (10pts)**

Create a new program called RotateString.py, and write the function RotateString(s,k) to take a string s and a possibly-negative integer k. If k is non-negative, the function returns the string s rotated k places to the left. If k is negative, the function returns the string s rotated |k| places to the right. For example:

*print(RotateString ('abcd', 1) == 'bcda')*

*print(RotateString ('abcd', -1) == 'dabc')*

Please show your test case in report and give some explanations about how do you solve it.

# **Exercise 3.1 ApplyCaesarCipher (10pts)**

A [Caesar Cipher](https://brilliant.org/wiki/caesar-cipher/#:~:text=A%20Caesar%20cipher%20is%20a%20simple%20method%20of,a%20Z%20as%20an%20A%2C%20and%20so%20on.) is a simple cipher that works by shifting each letter in the given message by a certain number. For example, if we shift the message "We Attack At Dawn" by 1 letter, it becomes "Xf Buubdl Bu Ebxo".

Create a new program ApplyCaesarCipher.py and write the function ApplyCaesarCipher(message, shift) which shifts the given message by shift letters. You are guaranteed that the message is a string, and that shift is an integer between -25 and 25. Capital letters should stay capital and lowercase letters should stay lowercase, and non-letter characters should not be changed. Note that "Z" wraps around to "A". So, for example:

*print(ApplyCaesarCipher("We Attack At Dawn", 1) == "Xf Buubdl Bu Ebxo")*

*print(ApplyCaesarCipher("zodiac", -2) == "xmbgya")*

Please show your test case in report and give some explanations about how do you solve it.

# **Exercise 3.2 SmallestDifference (10pts)**

Create a new program SmallestDifference.py and write the function SmallestDifference(a) that takes a list of integers and returns the smallest absolute difference between any two integers in the list. If the list has fewer than 2 elements, return -1. For example:

*print(SmallestDifference([19,2,83,6,27]) == 4)*

The two closest numbers in that list are 2 and 6, and their difference is 4.

Please draw diagram in the report.

# **Exercise 3.3 WordSearch (10pts)**

You are given an array of string products and a string searchWord. Create a new program WordSearch.py and design a system that suggests at most three product names from products after each character of searchword is typed. Suggested products should have a common prefix with searchword. If there are more than three products with a common prefix return the three lexicographically minimums products.

Return a list of lists of the suggested products after each character of searchWord is typed.

Example 1:

*Input: products = ["mobile","mouse","moneypot","monitor","mousepad"], searchWord = "mouse"*

*Output: [*

*["mobile","moneypot","monitor"],*

*["mobile","moneypot","monitor"],*

*["mouse","mousepad"],*

*["mouse","mousepad"],*

*["mouse","mousepad"]*

*]*

Example 2:

*Input: products = ["havana"], searchWord = "havana"*

*Output: [["havana"],["havana"],["havana"],["havana"],["havana"],["havana"]]*

Example 3:

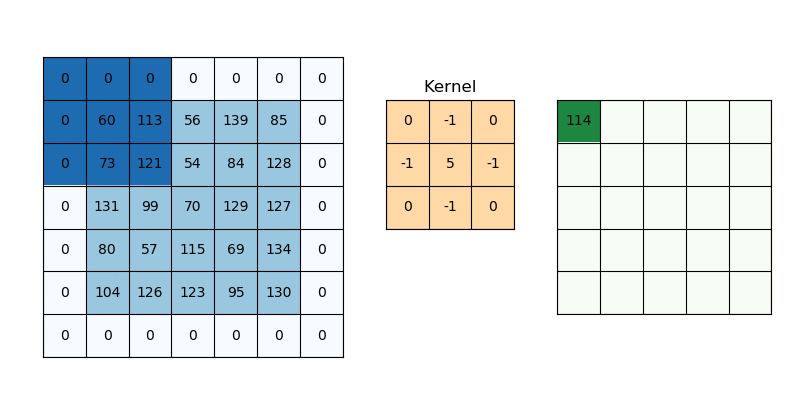
*Input: products = ["bags","baggage","banner","box","cloths"], searchWord = "bags"*

*Output: [["baggage","bags","banner"], ["baggage","bags","banner"], ["baggage","bags"], ["bags"]]*

# **Exercise 3.4 Image2DConvolution (10pts)**

Create a program called Image2dConv.py to realize the convolution operator between image and image kernel. Create a function Image2dConv(*img*, *kernel*) with an M\*M integer matrix *img* representing the grayscale of an image, an image *kernel* is a filter of the size n\*n (n <= M) that can be applied to each cell of an image by convolution operator (Please read [material about 2D convolution](http://www.songho.ca/dsp/convolution/convolution2d_example.html), and also see the example 1), return the image (M\*M integer matrix) after applying the kernel on each cell of it.

Example 1:



Example 2:

*>>>img = [[1,1,1],[1,0,1],[1,1,1]]*

*>>>kernel = [[1,1,1],[1,1,1],[1,1,1]]*

*>>>print(Image2dConv(img,kernel))*

*[[3,5,3],[5,8,5],[3,5,3]]*