Linear Arrays [CO5]

[Each method carries 5 marks]

Instructions for students:

- Complete the following methods on Arrays
- ONLY USE NUMPY ARRAY. DO NOT USE LIST.
- You may use any language to complete the tasks.
- All your methods must be written in one single .java or .py or .pynb file. DO NOT CREATE separate files for each task.
- If you are using JAVA, you must include the main method as well which should test your other methods and print the outputs according to the tasks.
- If you are using PYTHON, then follow the coding templates shared in this folder.

NOTE:

- **YOU CANNOT USE ANY BUILT-IN FUNCTION EXCEPT len IN PYTHON. [negative indexing, append is prohibited]**
- **YOU HAVE TO MENTION SIZE OF ARRAY WHILE INITIALIZATION**
- **YOUR CODE SHOULD WORK FOR ANY VALID INPUTS. [Make changes to the Sample Inputs and check whether your program works correctly]**
- **Use Numpy array to initialize the array. You are allowed to use the shape parameter of the numpy array as well.**
- **DO NOT USE DICTIONARY**

Linear Arrays [CO5]

[Each method carries 5 marks]

Array Tasks:

1. Merge Lineup:

Now you guys are playing a 2v2 fierce pokemon battle. At one point your and your teammate's pokemons have very low hp. So you created a new rule to merge the hp of your and your teammate's pokemons and create a new pokemon lineup. But the opposite team placed a special condition. You and your teammate have to add the hp of your team's pokemons from opposite directions. That is, your first pokemon's hp will be added to your teammate's last pokemon's hp; your second pokemon's hp will be added to your teammate's second last pokemon's hp and so on and so forth. The None elements denote 0 hp. You and your teammate have the same number of pokemons. Implement the scenario using a proper method which takes your and your teammate's pokemon hp as two arrays (as parameters); and returns the resulting arr.

Sample Input / Driver Code:
pokemon_1: [4, 5, -1, None, None]
pokemon_2: [2, 27, 7, 12, None]

#Function Call: print(mergeLineup(pokemon_1, pokemon_2))

Sample Output:
[4,17,6,27,2]

Explanation:
4+None(0) = 4 {pokemon_1[0]+pokemon_2[4]},
5+12 = 17 {pokemon_1[1]+pokemon_2[3]},
-1+7 = 6 {pokemon_1[2]+pokemon_2[2]},
None(0)+27 = 27 {pokemon_1[3]+pokemon_2[1]},
None(0)+2 = 2 {pokemon_1[4]+pokemon_2[0]}

Sample Input / Driver Code:
pokemon_1: [12, 3, 25, 1, None]
pokemon_2: [5, -9, 3, None, None]

Linear Arrays [CO5]

[Each method carries 5 marks]

#Function Call: print(mergeLineup(pokemon_1, pokemon_2))
Sample Output:
[12, 3, 28, -8, 5]

2. Discard Cards:

You and your friends are playing a card game. You pick n numbers of UNO cards from the deck. The rule of the card game is- one person says a number, **t** and you have to discard the **alternative** cards with the same number from your hand without changing the relative position of other cards. Implement the scenario using a proper method which takes the cards in your hands as an array and a number as parameters; and returns the resulting array. The empty spaces are denoted as 0 in the array.
Start from deleting the first t.

Sample Input / Driver Code:
cards=[1,3,7,2,5,2,2,2,0]
 	
#Function Call:
print(discardCards(cards, t = 2))
 	
Sample Output:
[1,3,7,5,2,2,0,0,0]
 	
Explanation:
[1,3,7,2_0,5,2_1,2_2,2_3,0]
Starting from the first 2, the alternative 2's are 2_0 and 2_2.
After removing them, the resulting array is -
[1,3,7,5,2,2,0,0,0]
	Sample Input / Driver Code:
	cards=[5,5,5,0,0]
	
	#Function Call:
	print(discardCards(cards, t = 5))
	
	Sample Output:
	[5,0,0,0,0]
	
	Explanation:
	[5_0, 5_1, 5_2,0,0]
	Starting from the first 5, the alternative 5's are 5_0 and 5_2.
	After removing them, the resulting array is -
	[5,0,0,0,0]

Linear Arrays [CO5]

[Each method carries 5 marks]

3. DUBER Fare Splitting:

Suppose, you and your whole group just finished the CSE220 class in BracU at 5PM. All of you are going to the Oppenheimer screening at Bashundhara. You know that now the buses will be the most congested, so you decide to go by DUBER. DUBER is a ride sharing app like UBER which can take **at most** two passengers. Each DUBER car has the same amount of fare but all of you have variable amounts of money with you. So now, you need to split up in groups of **at most** 2 so that the total amount of money of the group is equal to the fare of a DUBER car so that you can pay easily. Now given an array of the amount of money each of you have, write a program to find and print groups of maximum 2 that have a total amount of money exactly equal to the fare. If someone can not be grouped in such a way, **create an array with them and print the ungrouped array.**
[Printing the group numbers is optional]

Hint:

1. You do not need to create arrays of the eligible groups; print them inside the function only.
2. Group Member serial or group serial does not matter
3. You need to create an array of ungrouped members and print.

Sample Input	Sample Output
findGroups([120, 100, 150, 50, 30], 150)	Group 1 : 120, 30 Group 2 : 100, 50 Group 3 : 150
findGroups ([60, 150, 60, 30, 120, 30], 180)	Group 1 : 60, 120 Group 2 : 30, 150 Ungrouped : 30 60
findGroups ([30, 150, 150], 180)	Group 1 : 30, 150 Ungrouped: 150

Linear Arrays [CO5]

[Each method carries 5 marks]

4. Get Those Hobbies:

Imagine you are conducting a survey to gather information about people's preferences in a small town. Participants are asked to provide an array of their favorite activities or hobbies. As the survey responses come in, you want to analyze the unique activities that people in the town enjoy. Based on the scenario, design a proper method which takes people's hobbies as arrays and prints how many participants enjoy each hobby inside the method.

Hint: Your method should work for a variable number of arrays passed as parameters. You may or may not create a new array.

Sample Input / Driver Code:
participant_1 = ["Hiking", "Reading", "Photography", "Cooking"]
participant_2 = ["Reading", "Hiking", "Painting"]
participant_3 = ["Hiking", "Cooking", "Photography"]

#Function Call: print(analyzeHobbies(participant_1, participant_2, participant_3))

Sample Output:
Unique Activities in the Town:
['Photography', 'Painting', 'Cooking', 'Reading', 'Hiking']

Statistics:
2 participant(s) like(s) Photography.
1 participant(s) like(s) Painting.
2 participant(s) like(s) Cooking.
2 participant(s) like(s) Reading.
3 participant(s) like(s) Hiking.

Sample Input / Driver Code:
participant_1 = ["Gardening", "Traveling"]
participant_2 = ["Singing", "Gardening", "Painting"]

#Function Call: print(analyzeHobbies(participant_1, participant_2))

Linear Arrays [CO5]

[Each method carries 5 marks]

Sample Output:

Unique Activities in the Town:
[Gardening, Traveling, Singing, Painting]

Statistics:
2 participant(s) like(s) Gardening.
1 participant(s) like(s) Traveling.
1 participant(s) like(s) Singing.
1 participant(s) like(s) Painting.

Linear Arrays [CO5]

[Each method carries 5 marks]

BONUS UNGRADED TASK

Look and Say:

There is a particular sequence named ‘Look and Say’ sequence. The sequence begins with a single digit in which the next term is obtained by describing the previous term. For example, we start the sequence with number 1 and then we pronounce the number correctly. As for the number 1, It must be read as "one 1," where “one” represents the total digits of that number .We write the "one 1" numerically and we get the number 11. Next we can read 11 as "two 1." So the next number of this sequence is 21. We read 21 as “one 2, one 1”. So the next number of this sequence is 1211. Thus the sequence is: 1, 11, 21, 1211 [read as one 1, one 2, two 1], 111221, ...

You are given a number where the digits are the elements of a numpy array. Write a method that generates the next number in the look and say sequence and stores it in a new array.

Constraint: You cannot use any other datatype [i.e. string] or data structure except for arrays.

Hint: The size of the new array will never be more than 100.

[You need not worry about the extra zeroes at the end of your resulting array]

Sample Input	Sample Output	Explanation	Function Call
arr = [1,3,1,1,2,2,2,1]	result = [1,1,1,3,2,1,3,2,1,1]	[1,3,1,1,2,2,2,1] read as One 1, one 3, two 1, three 2, one 1. Thus the result is [1,1,1,3,2,1,3,2,1,1]	arr =np.array([1,3,1,1,2,2,2,1]) print(look_and_say(arr))
arr = [3,1,1,3]	result = [1,3,2,1,1,3]	[3,1,1,3] read as one 3, two 1, one 3. Thus the result is [1,3,2,1,1,3]	arr =np.array([3,1,1,3]) print(look_and_say(arr))