Part 3

Q1)Paraphrase the problem in our own words.

For this question, we are given integers and the task is to obtain or find the first number that shows up more than one time as you can scan left to right. If multiple numbers are repeated, we return the one whose second occurrence comes first in the list. If no duplicates appears than return -1.

Q2)Create 1 new example that demonstrates you understand the problem. Trace/walkthrough 1 example that your partner made and explain it.

my example Input: nums = [10, 5, 3, 5, 10, 8] Output : 5

Partner example Input: nums = [7, 8, 6, 9, 8, 9] Trace Step 1: 7->first time ->{7}

Step 2: 8->first time ->{7,8}

Step 3: 6->first time-> {7,8,6}

Step 4: 9->first time-> {7,8,6,9}

Step 5: 8-> already in set/duplicate-> return -1

Although 9 also appears later, the second occurrence of 8 comes first, so 8 is returned.

Q3) Copy the solution your partner wrote.

from typing import List

def first_duplicate(nums: List[int]) -> int:

```
seen_dict={}

for i in range(len(nums)):

    n=nums[i]

    if n in seen_dict:

        return n

    seen_dict[n]=i

return -1
```

g=first_duplicate(nums)

g

Q4)Explain why their solution works in your own words.

The function uses a dictionary so it can keep track of numbers that already have appeared within the list. It will iterate through each element and if a number is within the dictionary, it then will return its first duplicate

If the loop finishes without finding one it will return -1.

Q5) Explain the problem’s time and space complexity in your own words.

The time complexity for this would be o(n). Due to the fact that, each element processed once, and dictionary lookups take constant time on average.

The space complexity for this would be o(n). The reason for this as worst case , the dictionary stores every element in the list, basically meaning worst case no duplicates.

Q6) Critique your partner's solution, including explanation, and if there is anything that should be adjusted.

The partner's solution has strengths & areas of improvement (A.O.I)

Strenght: efficient, simple, easy-logic, and early return stops processing as soon as duplicate is found. A.O.I: Using set instead of dictionary & proper return statement i.e (print(g))

Part 4

Q1) Please write a 200 word reflection documenting your process from assignment 1, and your presentation and review experience with your partner at the bottom of the Jupyter Notebook under a new heading "Reflection." Again, export this Notebook as pdf.

Reflection I was able to improve my knowledge of stacks and Python algorithmic problem-solving by working on Question Two, Valid Bracket Sequence. I tackled the issue by realizing that a stack's last-in, first-out structure makes it perfect for tracking the most recent unmatched opening brackets. I started my solution with an empty stack and a dictionary that matched each closing bracket to its matching opening bracket. I pushed opening brackets onto the stack as I went along the string. When I came across a closing bracket, I carefully removed the top element from the stack—using a placeholder if the stack was empty—and compared it to the mapping's predicted opening bracket. The function returned False right away if the brackets did not match. At the end, I returned True if the stack was empty, indicating all brackets were correctly matched.Implementing this solution required careful attention to edge cases, such as empty strings, unmatched closing brackets, and nested sequences. I also focused on writing clean, readable code with descriptive variable names and comments to make the logic transparent. Through this assignment, I reinforced my ability to design and implement stack-based algorithms efficiently, ensuring both correctness and clarity in my solution.