Paraphrase the problem in your own words.

My partner answered Q3. You have a list of integers ranging from 0 to n, where n is the length of the list. Your task is to identify and provide a list of numbers that are absent within this range. If the list contains all numbers from 0 to n, return -1. It's important to consider that the list may have non-unique integers.

Create 1 new example that demonstrates you understand the problem.

```
input_list = [1, 3, 0, 2, 2, 5, 4]
#In this case, the length of the list is 7 (n=7). The missing numbers
within the range [0, 7] are 6 and 7.
#Therefore, the expected output would be:
output_result = [6, 7]
```

#Trace/walkthrough 1 example that your partner made and explain it.

```
#In my partners example:

Input: [3, 1, 0, 4, 6]

#The length of the list is 5 (n=5). The missing numbers within the
range [0, 5] are 2 and 5. Therefore, the expected output is:

Output: [2, 5]

#These are the numbers that are absent in the given list within the
specified range [0, 5].
```

Copy the solution your partner wrote.

```
from typing import List
def missing_num(nums: List[int]) -> List[int]:
    if not nums: return [-1]
    n = max(nums) # Get the maximum number in the list
    num_set = set(nums)
    missing_numbers = []
    for i in range(n+1):
        if i not in num_set:
            missing_numbers.append(i)
    return missing_numbers if missing_numbers else [-1]
```

```
print(missing_num([0, 2])) # Output: [1]
print(missing_num([5, 0, 1])) # Output: [2, 3, 4]
print(missing_num([6, 8, 2, 3, 5, 7, 0, 1, 10])) # Output: [4, 9]
print(missing_num([3, 1, 0, 4, 6]))
print(missing_num([9, 8, 7, 6, 5, 4, 3, 2, 1, 0]))
```

```
[1]
[2, 3, 4]
[4, 9]
[2, 5]
[-1]
```

Explain why their solution works in your own words.

(my explanation via comments below)

```
from typing import List
def missing_num(nums: List[int]) -> List[int]:
    if not nums: return [-1] #If the input list nums is empty, it
returns [-1] to indicate that there are no missing numbers. This
covers the case where the list is empty.
    n = max(nums) #It finds the maximum number in the list nums using
the max() function. This maximum number, denoted as n, helps determine
the range of numbers to consider for missing values.
    num_set = set(nums)

    #t iterates through the range [0, n] and checks if each number is
present in the set num_set.
    #If a number is not in the set, it means it is missing in the
original list, and it is added to the missing_numbers list.
    missing_numbers = []
    for i in range(n+1):
        if i not in num_set:
            missing_numbers.append(i)
    return missing_numbers if missing_numbers else [-1] #If there are
missing numbers, it returns the list of missing numbers. If no missing
numbers are found, it returns [-1] to signify that all numbers within
the specified range are present in the original list.

print(missing_num([0, 2])) # Output: [1]
print(missing_num([5, 0, 1])) # Output: [2, 3, 4]
print(missing_num([6, 8, 2, 3, 5, 7, 0, 1, 10])) # Output: [4, 9]
print(missing_num([3, 1, 0, 4, 6]))
print(missing_num([9, 8, 7, 6, 5, 4, 3, 2, 1, 0]))
```

#Explain the problem's time and space complexity in your own words.

the time complexity is O(n) due to the linear iteration through the range, and the space complexity is also O(n) due to the set and list that may grow with the size of the input list and range.

#Critique your partner's solution, including explanation, if there is anything should be adjusted.

```
#here is an improved version

from typing import List

def missing_num(nums: List[int]) -> List[int]:
    if not nums:
        return [-1]

    num_set = set(nums)
    missing_numbers = list(set(range(len(nums))) - num_set)

    return missing_numbers if missing_numbers else [-1]
```

In this improved version:

- The maximum number is not explicitly calculated.
- The range is determined directly from the length of the input list.
- Set difference is used to efficiently find missing numbers.
- The need for a separate list is eliminated.