Job Interview Questions: Array

Question1: Assume that you are given an integer array sorted in non-decreasing order, write a function (`remove_duplicates_from_sorted_array`) to remove the duplicates in this array such that each unique element appears only once (The relative order of the elements should be kept the same). The function should return k (the number of elements after removing the duplicate) as its return. Please Mention the com[utational Complexity of your solution

Note: This should be an in-place solution, meaning that you can not use any extra memory and you should write to the same input array, so the **space complexity** must be O(1).

Example input/output:

nums = [0,0,1,1,1,2,2,3,3,4,5,5]

`remove_duplicates_from_sorted_array(nums)`

Output: 6, nums = [0,1,2,3,4,5,_,_,_,_,_,_]

The values after the first k element do not matter, in fact the nums array will be as:

[0,1,2,3,4,5,2,3,3,4,5,5]

Solution:

We will use two pointers i and j. As long as nums[i] = nums[j], we increment j and skip the duplicate until we reach a nums[j] != nums[i], which means we have a new value and we should copy its value to nums[i + 1]:

```
def remove_duplicates_from_sorted_array(nums):
    if (len(nums) == 0) or (nums is null) return 0;
    i=0
    for j in range(1,len(nums)):
        if nums[j]!=nums[i]:
            i+=1
            nums[i]=nums[j]
    return i+1
```

Complexity analysis:

- Time complexity : $O(n)$ (n is the length of the array)
- Space complexity : $O(1)$

Question2:

A palindrome is a word or phrase that reads the same backwards as forwards, for example, radar, level, rotor, kayak, reviver, racecar, madam, and refer. Implement a function is_palindrome(x) using a stack that returns true if a string is palindrome and false otherwise (assume you have access to a stack implementation).

Solution:

```
from queue import LifoQueue
# Initializing a stack


def is_palindrome(word: str) -> bool:
   length = len(word)
   mid = length // 2
   stack = LifoQueue(maxsize = mid)
   i = 0
   while i < mid:
       stack.put(word[i])
       i += 1


  # if the length of the string is odd, neglect the middle character
   if length % 2 != 0:
```

```
        i += 1
    while i < length:
        e = stack.get()
        if e != word[i]:
            return False
        i += 1
    return True

# Driver Code
if __name__ == "__main__":
    string = "madam"
    if is_palindrome(string):
        print("It is a palindrome!")
    else:
        print("No, It is not a palindrome!")
```

Question 3:

Implement a queue (enqueue/dequeue function) using (only) two stacks

Solution:

```
stack1=initialize_stack()
stack2=initialize_stack()

void enqueue(x) {
  push (stack1,x);
}

item dequeue(){
```

```
  if(is_empty(stack2)) then
    if(is_empty(stack1)) then {
       print("stack is empty!");
       return;
    }
    else while (!(is_empty(stack1))){
       x=pop(stack1);
       push(stack2,x);
    }
 return pop(stack2);

}
```