Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam - 603 110 (An Autonomous Institution, Affiliated to Anna University, Chennai)

UCS2403: DESIGN & ANALYSIS OF ALGORITHMS Assignment 3

1. Given a list L of n numbers, an inversion is defined as a pair (L[i],L[j]) such that i < j and L[i] > L[j]. For example, if L = [3,2,8,1], then (3, 2), (8, 1), (2, 1), (3, 1) are the inversions in L.

   - (a) Consider the Python codes given in (1) and (2) below for finding the count of inversions in a list.

   (1)
```
def count_inversions1(nums):
    count = 0

    for i in range(1, len(nums)):
        if nums[i] < nums[i - 1]:

            count += 1
    return count
```

   (2)
```
def count_inversions2(nums):
    nums.sort()

    count = 0
    for i in range(1, len(nums)):

        if nums[i] < nums[i - 1]:
            count += 1

return count
```

   Both the given codes have errors. Find one counterexample for each of them. Recall that a counterexample is an input instance to the algorithm that produces a wrong output.

   - (b) Write your own (correct) code to find the count of inversions in a list.

Ex no: 3	Assignment-3	Name: M.Rohith
Date: 15-03-2023	Finding errors in code	3122 21 5001 085

A) Incorrect Program code:

```
def count_inversions1(nums):
    count = 0
    for i in range(1, len(nums)):
        if nums[i] < nums[i - 1]:
            count += 1
    return count
nums=[7,8,3,1,5]
print("The iversions count: ",count_inversions1(nums))
```

Output:

The number of inversions in the given list is 7, but the output obtained is 2.

Incorrect Program code:

```
def count_inversions2(nums):
    nums.sort()
    count = 0
    for i in range(1, len(nums)):
        if nums[i] < nums[i - 1]:
            count += 1
    return count
nums=[7,8,3,1,5]
print("The iversions count: ",count_inversions2(nums))
```

Output:

The number of inversions in the given list is 7, but the output obtained is 0.

Correct Program code:

```
def count_inversion1(nums):
    count=0
    for i in range(1,len(nums)):
        for j in range(i,-1,-1):
            if(nums[i]<nums[j]):
                count+=1
    return count

nums=[7,8,3,1,5]
print(count_inversion1(nums))
```

Output:

This is the correct output as the inversions in the given list [7,8,3,1,5] are [(7,3),(7,1), (7,5),(8,3),(8,1),(8,5),(3,1)] and hence the count is 7.

2. Given a list of integers, the comparison count sorting algorithm sorts the list as follows: For each integer at index i in the list, count the number of integers that are strictly less than it. In the sorted list, place the integer at the index equal to the number of integers that are less than it. For example, there are no integers less than the minimum integer in the list, so the minimum integer is placed at index 0.

Now, consider the code given below to sort a list of numbers using com- parison count sort.

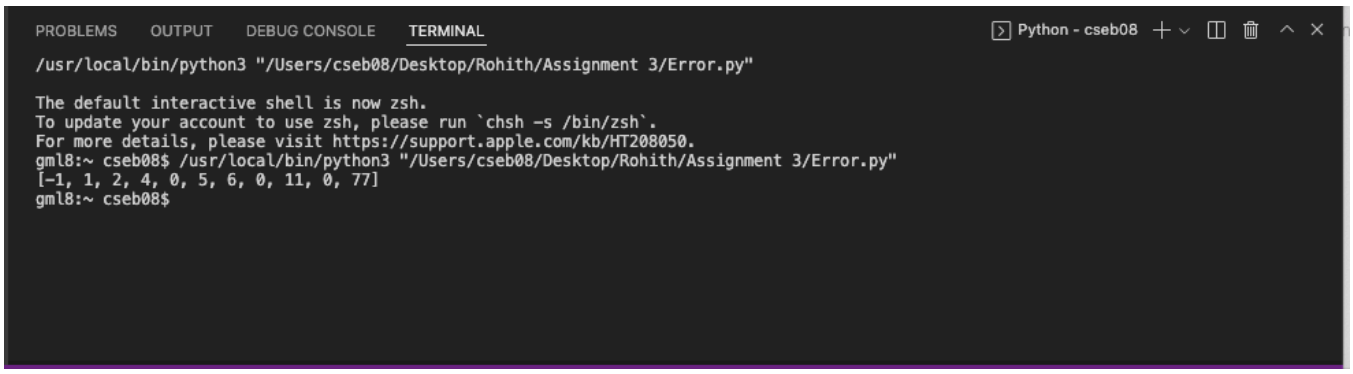
```
def comparison_count_sort(nums):
    count = [0] * len(nums)
    nums_sorted = [0] * len(nums)
    for i in range(len(nums) - 1):
        for j in range(i + 1, len(nums)):
            if nums[i] > nums[j]:
                count[i] += 1
            elif nums[i] < nums[j]:
                count[j] += 1
    for i in range(len(nums)):
        nums_sorted[count[i]] = nums[i]
    return nums_sorted
```

Find a counterexample to show that this code has errors. List the lines of code that have errors.

Incorrect Program code:

```
def comparison_count_sort(nums):
    count = [0] * len(nums)
    nums_sorted = [0] * len(nums)
    for i in range(len(nums) - 1):
        for j in range(i + 1, len(nums)):
            if nums[i] > nums[j]:
                count[i] += 1
            elif nums[i] < nums[j]:
                count[j] += 1
    for i in range(len(nums)):
        nums_sorted[count[i]] = nums[i]
    return nums_sorted
print(comparison_count_sort([11,2,4,1,11,5,6,77,6,4,-1]))
```

Output:



When numbers in the list are repeated, the repeated numbers in the sorted list are replaced with 0.

Correct Program code:

```
def comparison_count_sort(nums):
        count = [0] * len(nums)
        nums_sorted = [0] * len(nums)
        for i in range(len(nums) - 1):
            for j in range(i + 1, len(nums)):
                if nums[i] > nums[j]:
                    count[i] += 1
                elif nums[i] < nums[j]:
                    count[j] += 1
        for i in range(len(nums)):
            nums_sorted[count[i]] = nums[i]
        for i in range(len(nums_sorted)):
         if nums_sorted[i]==0:
                 nums_sorted[i]=nums_sorted[i-1]
        return nums_sorted
print(comparison_count_sort([11,2,4,1,11,5,6,77,6,4,-1]))
```

Output:

In order to get the correct output, we replaced the 0 with the previous element in the sorted list.