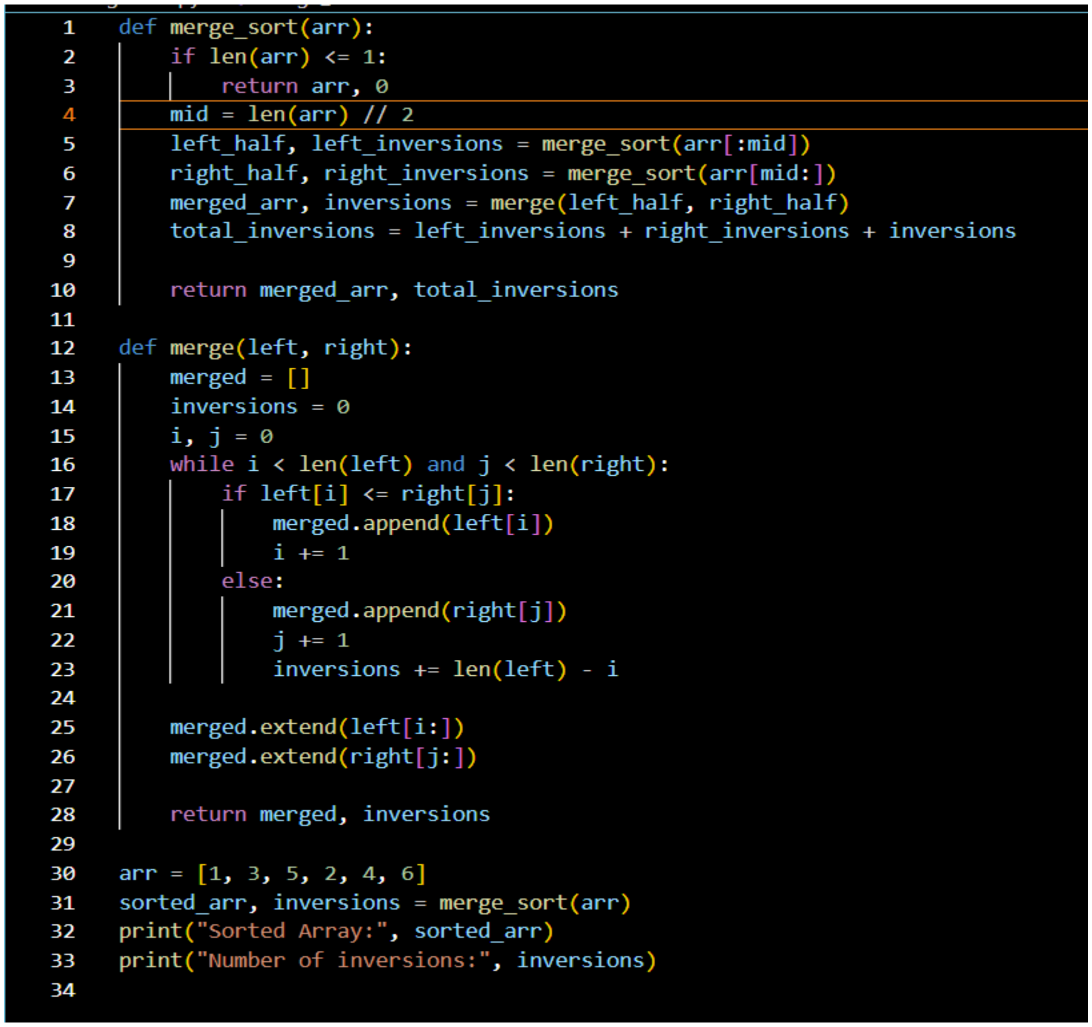
Abbottabad University of science and technology

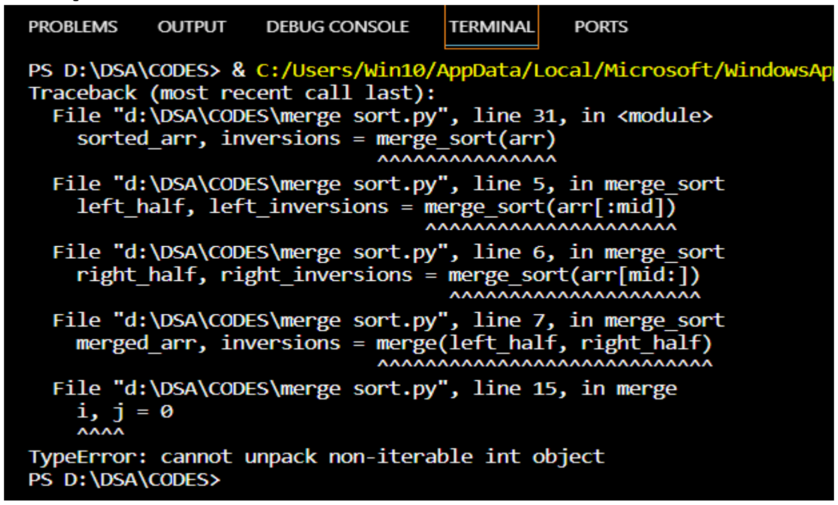
Department name :	Software Engineering
Submitted By :	Zakria
Roll No :	12415
Semester :	BSSE 3rd
Section :	'C'
Submitted To :	Jamal Abdul Ahad
Date :	06/11/23
Subject :	DSA
Assignment :	04

Question No 1:



```
 1  def merge_sort(arr):
 2      if len(arr) <= 1:
 3          return arr, 0
 4      mid = len(arr) // 2
 5      left_half, left_inversions = merge_sort(arr[:mid])
 6      right_half, right_inversions = merge_sort(arr[mid:])
 7      merged_arr, inversions = merge(left_half, right_half)
 8      total_inversions = left_inversions + right_inversions + inversions
 9
10      return merged_arr, total_inversions
11
12  def merge(left, right):
13      merged = []
14      inversions = 0
15      i, j = 0
16      while i < len(left) and j < len(right):
17          if left[i] <= right[j]:
18              merged.append(left[i])
19              i += 1
20          else:
21              merged.append(right[j])
22              j += 1
23              inversions += len(left) - i
24
25      merged.extend(left[i:])
26      merged.extend(right[j:])
27
28      return merged, inversions
29
30  arr = [1, 3, 5, 2, 4, 6]
31  sorted_arr, inversions = merge_sort(arr)
32  print("Sorted Array:", sorted_arr)
33  print("Number of inversions:", inversions)
34
```

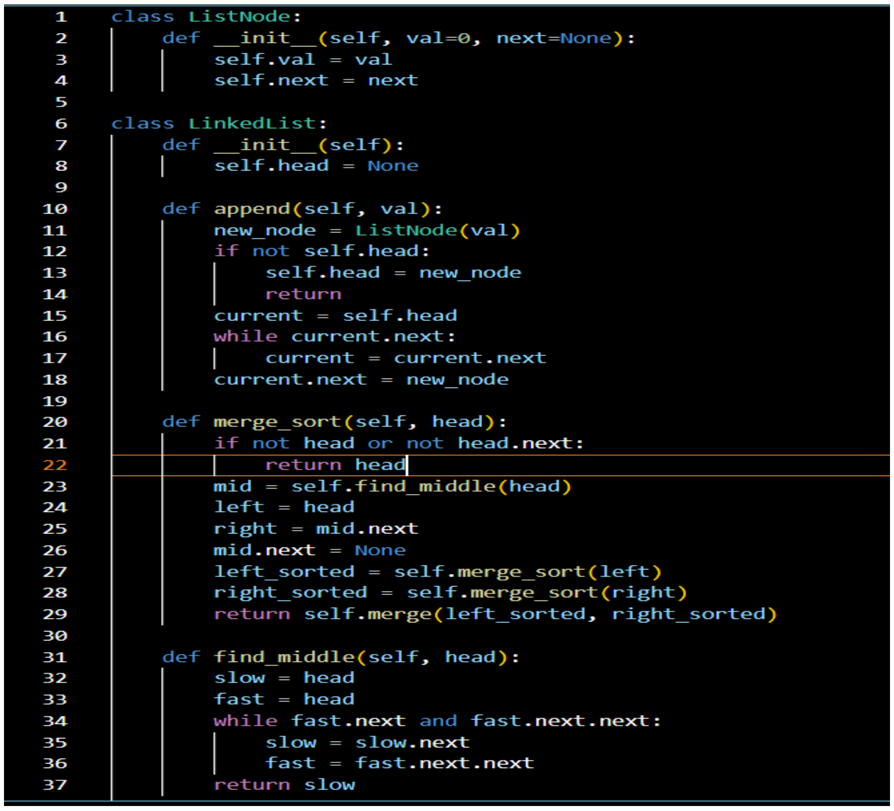
Output:



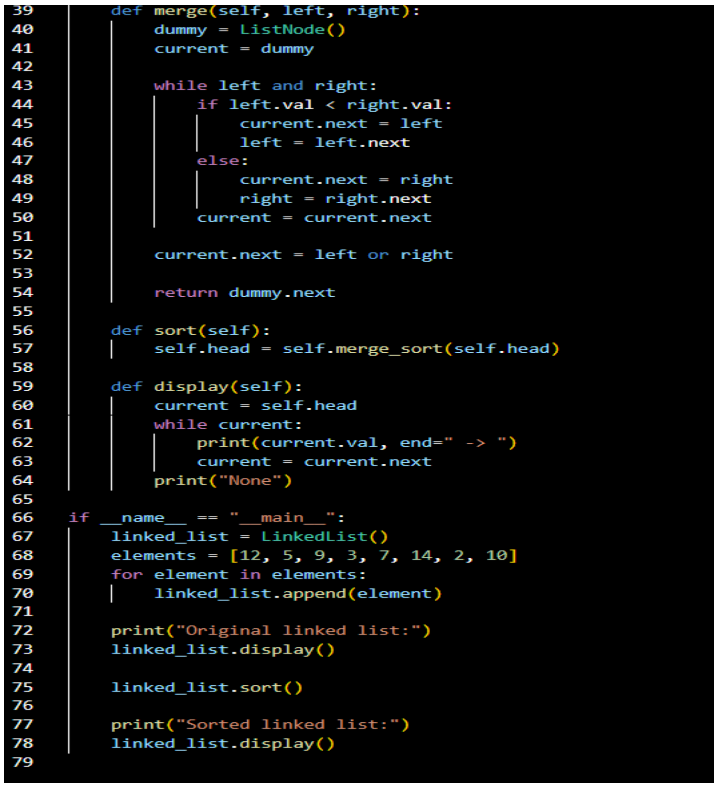
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS D:\DSA\CODES> & C:/Users/Win10/AppData/Local/Microsoft/WindowsAp
Traceback (most recent call last):
  File "d:\DSA\CODES\merge sort.py", line 31, in <module>
    sorted_arr, inversions = merge_sort(arr)
                             ^^^^^^^^^^^^^^^
  File "d:\DSA\CODES\merge sort.py", line 5, in merge_sort
    left_half, left_inversions = merge_sort(arr[:mid])
                                 ^^^^^^^^^^^^^^^^^^^^^
  File "d:\DSA\CODES\merge sort.py", line 6, in merge_sort
    right_half, right_inversions = merge_sort(arr[mid:])
                                   ^^^^^^^^^^^^^^^^^^^^^
  File "d:\DSA\CODES\merge sort.py", line 7, in merge_sort
    merged_arr, inversions = merge(left_half, right_half)
                             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "d:\DSA\CODES\merge sort.py", line 15, in merge
    i, j = 0
    ^^^^
TypeError: cannot unpack non-iterable int object
PS D:\DSA\CODES>
```

Question No 2:

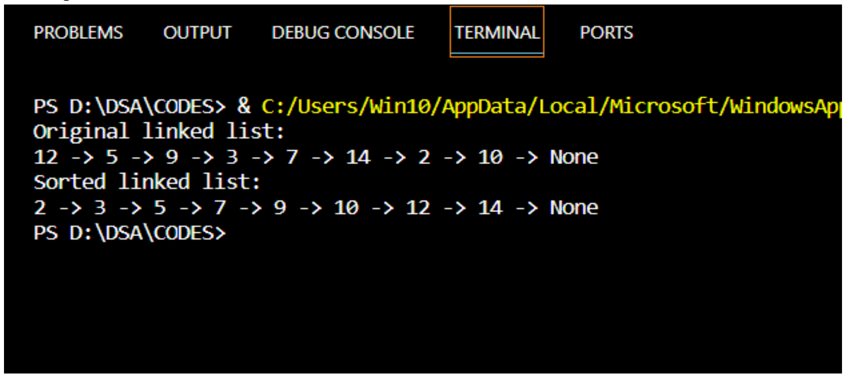


```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class LinkedList:
    def __init__(self):
        self.head = None

    def append(self, val):
        new_node = ListNode(val)
        if not self.head:
            self.head = new_node
            return
        current = self.head
        while current.next:
            current = current.next
        current.next = new_node

    def merge_sort(self, head):
        if not head or not head.next:
            return head
        mid = self.find_middle(head)
        left = head
        right = mid.next
        mid.next = None
        left_sorted = self.merge_sort(left)
        right_sorted = self.merge_sort(right)
        return self.merge(left_sorted, right_sorted)

    def find_middle(self, head):
        slow = head
        fast = head
        while fast.next and fast.next.next:
            slow = slow.next
            fast = fast.next.next
        return slow
```



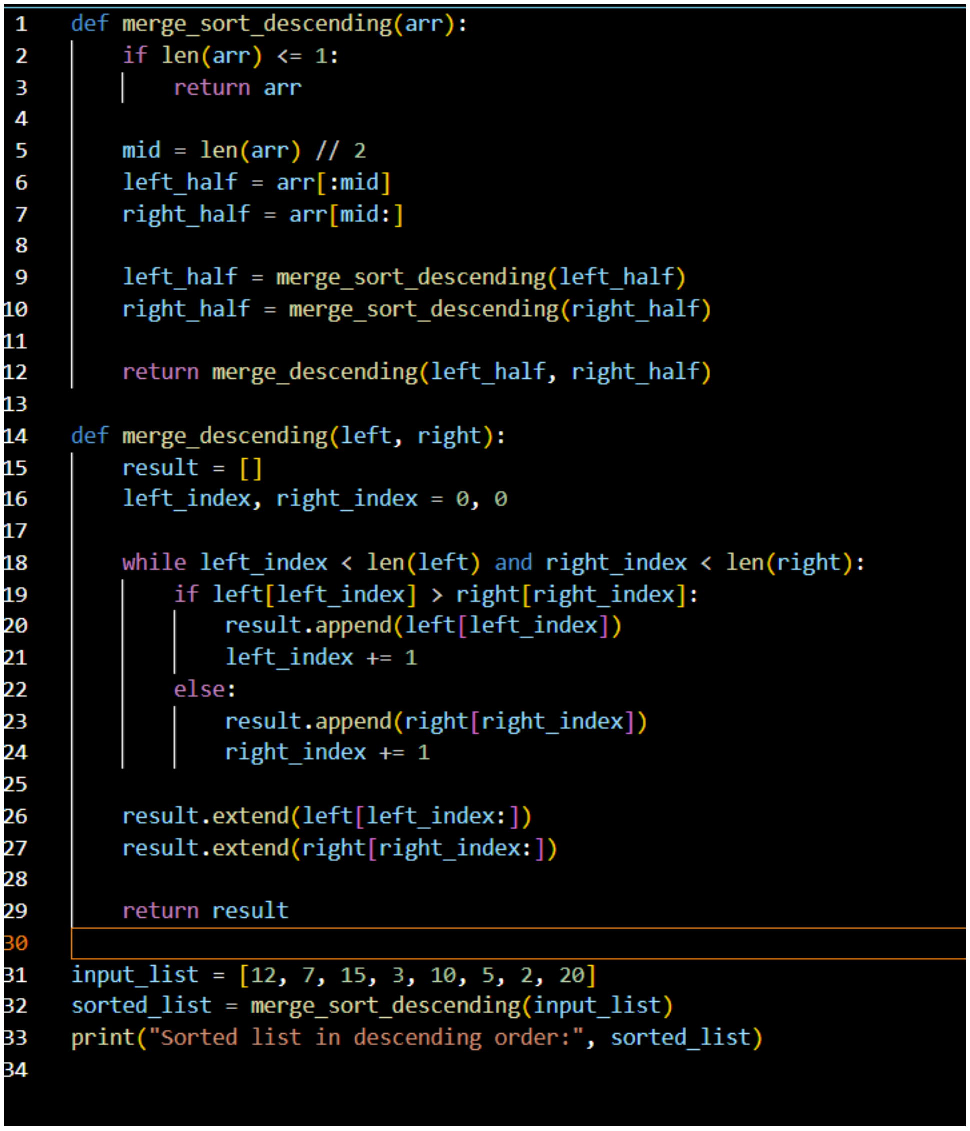
```python
    def merge(self, left, right):
        dummy = ListNode()
        current = dummy

        while left and right:
            if left.val < right.val:
                current.next = left
                left = left.next
            else:
                current.next = right
                right = right.next
            current = current.next

        current.next = left or right

        return dummy.next

    def sort(self):
        self.head = self.merge_sort(self.head)

    def display(self):
        current = self.head
        while current:
            print(current.val, end=" -> ")
            current = current.next
        print("None")

if __name__ == "__main__":
    linked_list = LinkedList()
    elements = [12, 5, 9, 3, 7, 14, 2, 10]
    for element in elements:
        linked_list.append(element)

    print("Original linked list:")
    linked_list.display()

    linked_list.sort()

    print("Sorted linked list:")
    linked_list.display()

```

Output:



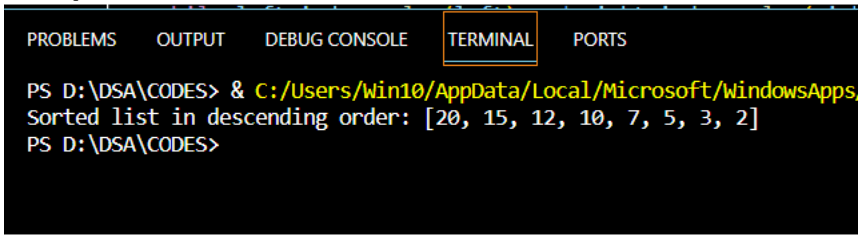
```
PS D:\DSA\CODES> & C:/Users/Win10/AppData/Local/Microsoft/WindowsAp
Original linked list:
12 -> 5 -> 9 -> 3 -> 7 -> 14 -> 2 -> 10 -> None
Sorted linked list:
2 -> 3 -> 5 -> 7 -> 9 -> 10 -> 12 -> 14 -> None
PS D:\DSA\CODES>
```

Question No 3:



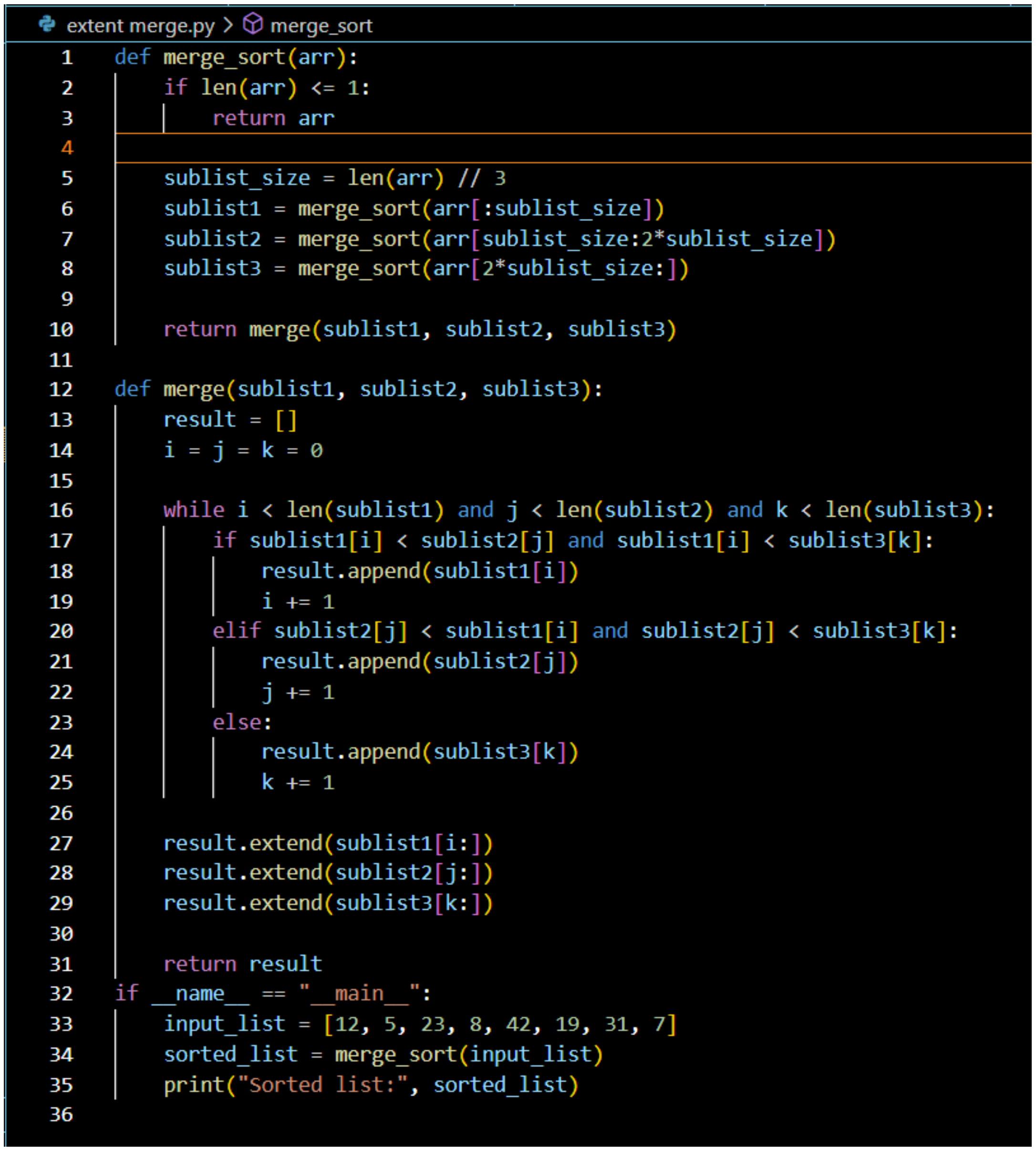
```python
def merge_sort_descending(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    left_half = merge_sort_descending(left_half)
    right_half = merge_sort_descending(right_half)

    return merge_descending(left_half, right_half)

def merge_descending(left, right):
    result = []
    left_index, right_index = 0, 0

    while left_index < len(left) and right_index < len(right):
        if left[left_index] > right[right_index]:
            result.append(left[left_index])
            left_index += 1
        else:
            result.append(right[right_index])
            right_index += 1

    result.extend(left[left_index:])
    result.extend(right[right_index:])

    return result

input_list = [12, 7, 15, 3, 10, 5, 2, 20]
sorted_list = merge_sort_descending(input_list)
print("Sorted list in descending order:", sorted_list)
```

Output:



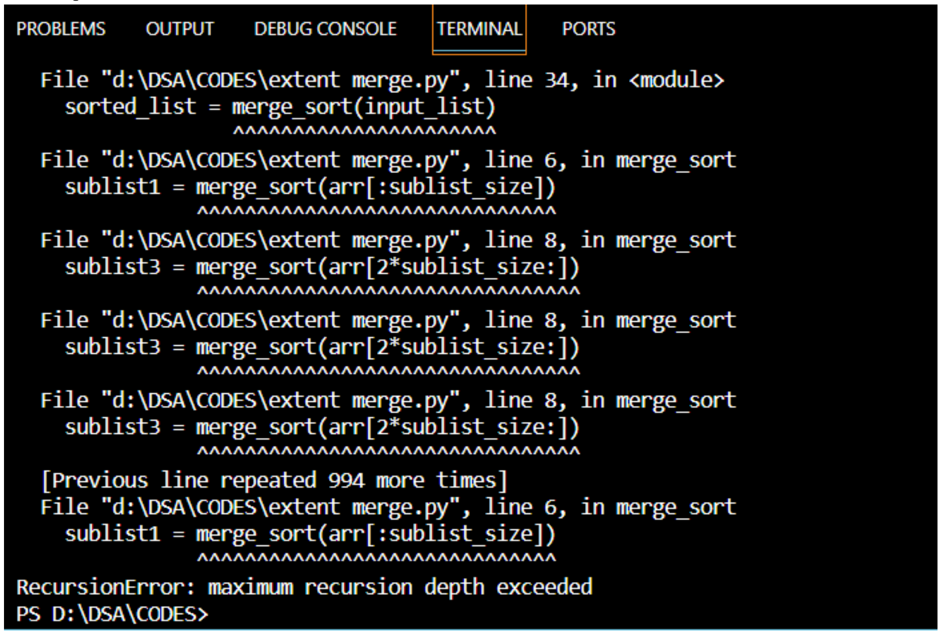
```
PS D:\DSA\CODES> & C:/Users/Win10/AppData/Local/Microsoft/WindowsApps
Sorted list in descending order: [20, 15, 12, 10, 7, 5, 3, 2]
PS D:\DSA\CODES>
```

Question No 4:



```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    sublist_size = len(arr) // 3
    sublist1 = merge_sort(arr[:sublist_size])
    sublist2 = merge_sort(arr[sublist_size:2*sublist_size])
    sublist3 = merge_sort(arr[2*sublist_size:])

    return merge(sublist1, sublist2, sublist3)

def merge(sublist1, sublist2, sublist3):
    result = []
    i = j = k = 0

    while i < len(sublist1) and j < len(sublist2) and k < len(sublist3):
        if sublist1[i] < sublist2[j] and sublist1[i] < sublist3[k]:
            result.append(sublist1[i])
            i += 1
        elif sublist2[j] < sublist1[i] and sublist2[j] < sublist3[k]:
            result.append(sublist2[j])
            j += 1
        else:
            result.append(sublist3[k])
            k += 1

    result.extend(sublist1[i:])
    result.extend(sublist2[j:])
    result.extend(sublist3[k:])

    return result
if __name__ == "__main__":
    input_list = [12, 5, 23, 8, 42, 19, 31, 7]
    sorted_list = merge_sort(input_list)
    print("Sorted list:", sorted_list)

```

Output:



```
  File "d:\DSA\CODES\extent merge.py", line 34, in <module>
    sorted_list = merge_sort(input_list)
                  ^^^^^^^^^^^^^^^^^^^^^^
  File "d:\DSA\CODES\extent merge.py", line 6, in merge_sort
    sublist1 = merge_sort(arr[:sublist_size])
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "d:\DSA\CODES\extent merge.py", line 8, in merge_sort
    sublist3 = merge_sort(arr[2*sublist_size:])
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "d:\DSA\CODES\extent merge.py", line 8, in merge_sort
    sublist3 = merge_sort(arr[2*sublist_size:])
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "d:\DSA\CODES\extent merge.py", line 8, in merge_sort
    sublist3 = merge_sort(arr[2*sublist_size:])
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  [Previous line repeated 994 more times]
  File "d:\DSA\CODES\extent merge.py", line 6, in merge_sort
    sublist1 = merge_sort(arr[:sublist_size])
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
RecursionError: maximum recursion depth exceeded
PS D:\DSA\CODES>
```