

**ABBOTTABAD UNIVERSITY OF SCIENCE AND TECHNOLOGY ABBOTTABAD**

**BS-SE (3) C**

**NAME : SAYYED ZAIN UL ABIDIN SHAH**

**SUBMITTED TO: SIR JAMAL**

**ASSIGNMENT : 01**

**SUBJECT : DSA**

**ROLL NO : 12387**

**DATE OF SUBMISSION: 5 NOV, 2023**

**Question no 1 :**

def merge\_sort(arr):

    if len(arr) <= 1:

        return arr, 0

    mid = len(arr) // 2

    left, left\_inversions = merge\_sort(arr[:mid])

    right, right\_inversions = merge\_sort(arr[mid:])

    merged, split\_inversions = merge(left, right)

    total\_inversions = left\_inversions + right\_inversions + split\_inversions

    return merged, total\_inversions

def merge(left, right):

    merged = []

    left\_idx, right\_idx = 0, 0

    inversions = 0

    while left\_idx < len(left) and right\_idx < len(right):

        if left[left\_idx] <= right[right\_idx]:

            merged.append(left[left\_idx])

            left\_idx += 1

        else:

            merged.append(right[right\_idx])

            right\_idx += 1

            inversions += len(left) - left\_idx

    merged.extend(left[left\_idx:])

    merged.extend(right[right\_idx:])

    return merged, inversions

def count\_inversions(arr):

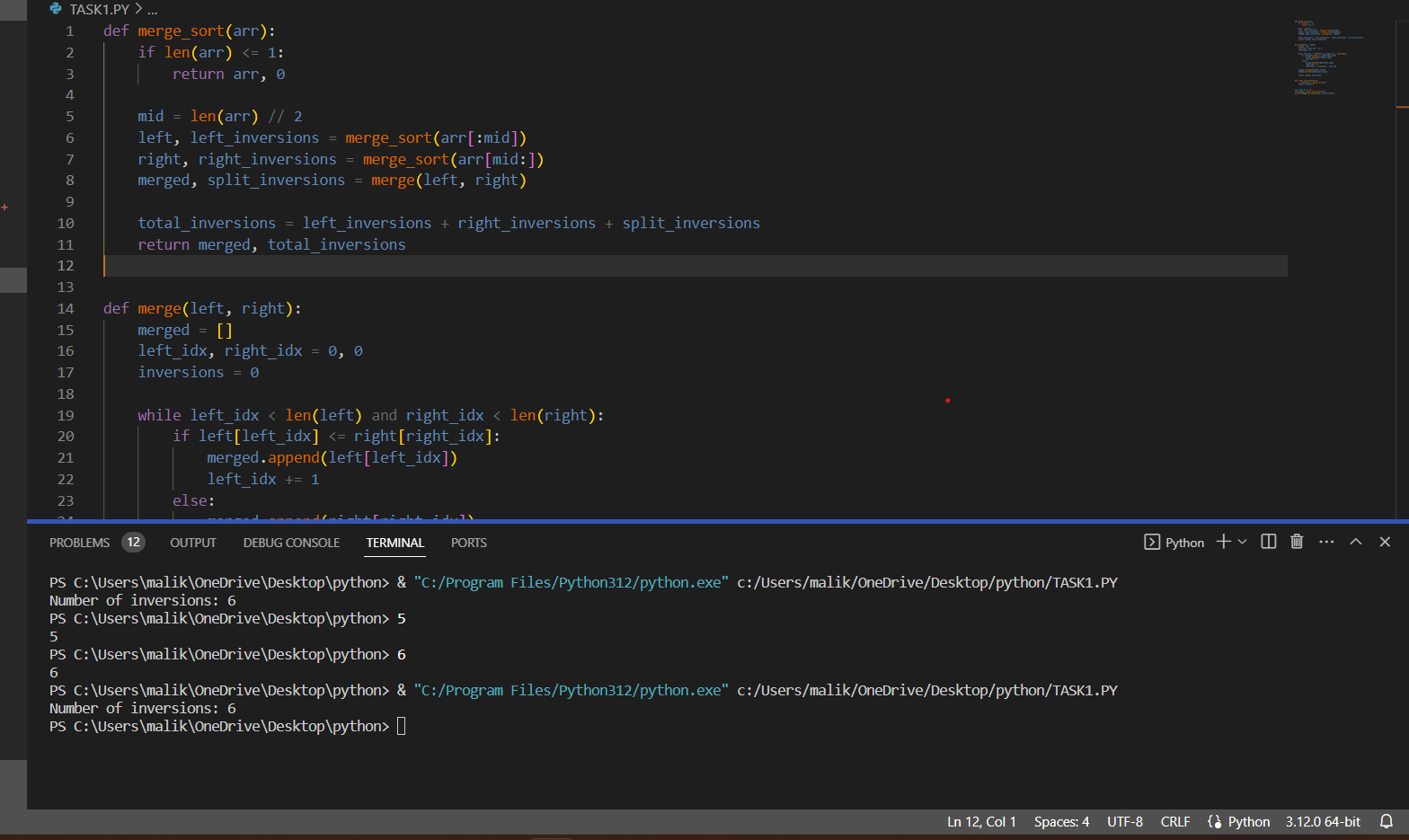
    \_, inversions = merge\_sort(arr)

    return inversions

arr = [8, 4, 2, 1]

inversions = count\_inversions(arr)

print(f"Number of inversions: {inversions}")



Question 02 class ListNode:

    def \_init\_(self, val=0, next=None):

        self.val = val

        self.next = next

def merge\_sorted\_lists(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 merge\_sort\_linked\_list(head):

    if not head or not head.next:

        return head

    # Split the linked list into two halves

    middle = find\_middle(head)

    left, right = head, middle.next

    middle.next = None

    left = merge\_sort\_linked\_list(left)

    right = merge\_sort\_linked\_list(right)

    return merge\_sorted\_lists(left, right)

def find\_middle(head):

    slow = head

    fast = head

    while fast.next and fast.next.next:

        slow = slow.next

        fast = fast.next.next

    return slow

def print\_linked\_list(head):

    current = head

    while current:

        print(current.val, end=" -> ")

        current = current.next

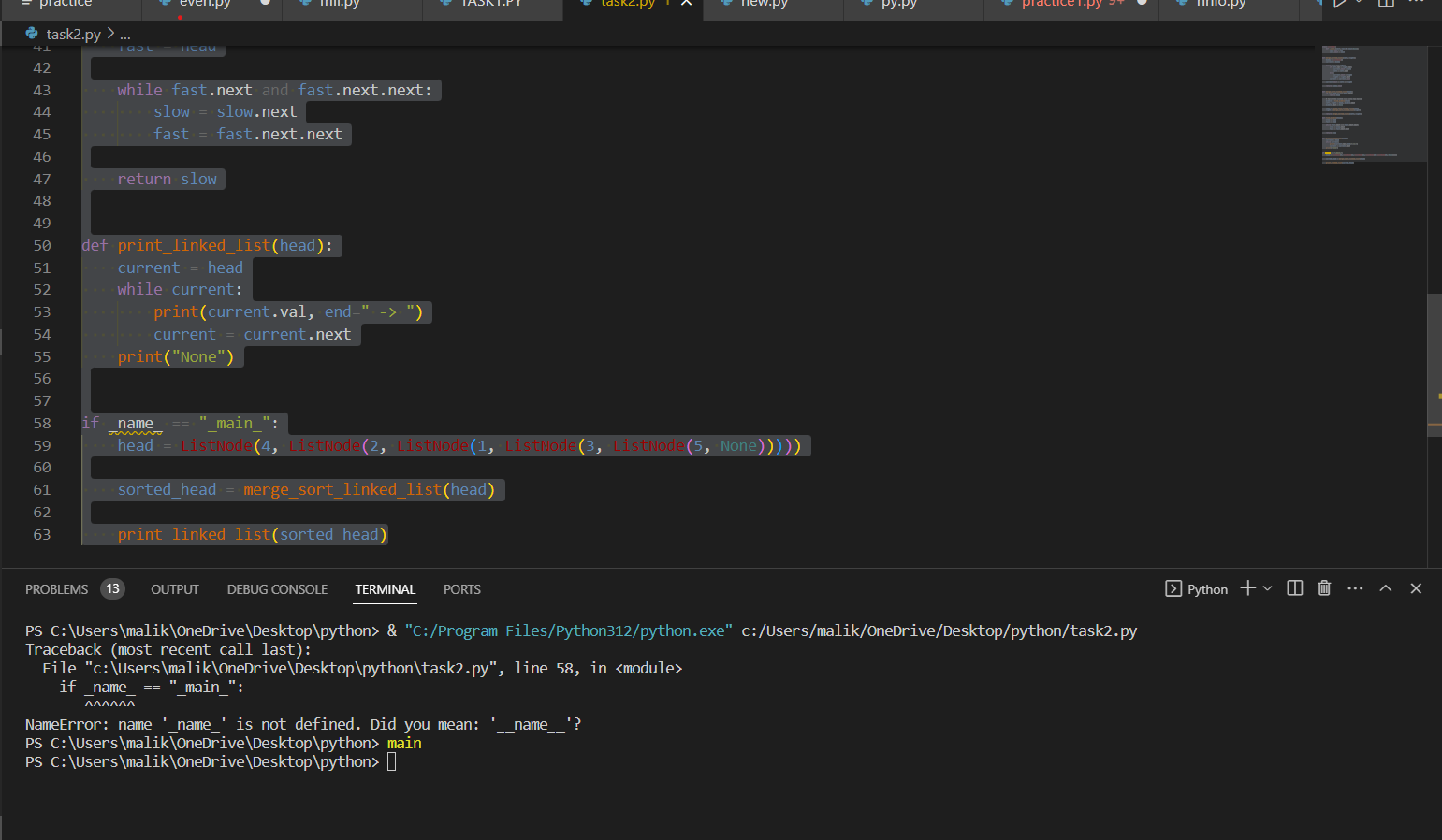
    print("None")

if \_name\_ == "\_main\_":

    head = ListNode(4, ListNode(2, ListNode(1, ListNode(3, ListNode(5, None)))))

    sorted\_head = merge\_sort\_linked\_list(head)

    print\_linked\_list(sorted\_head)



Question 03:

def merge\_sorted\_lists\_descending(left, right):

    merged = []

    i = j = 0

    while i < len(left) and j < len(right):

        if left[i] >= right[j]:

            merged.append(left[i])

            i += 1

        else:

            merged.append(right[j])

            j += 1

    merged.extend(left[i:])

    merged.extend(right[j:])

    return merged

def merge\_sort\_descending(arr):

    if len(arr) <= 1:

        return arr

    middle = len(arr) // 2

    left = arr[:middle]

    right = arr[middle:]

    left = merge\_sort\_descending(left)

    right = merge\_sort\_descending(right)

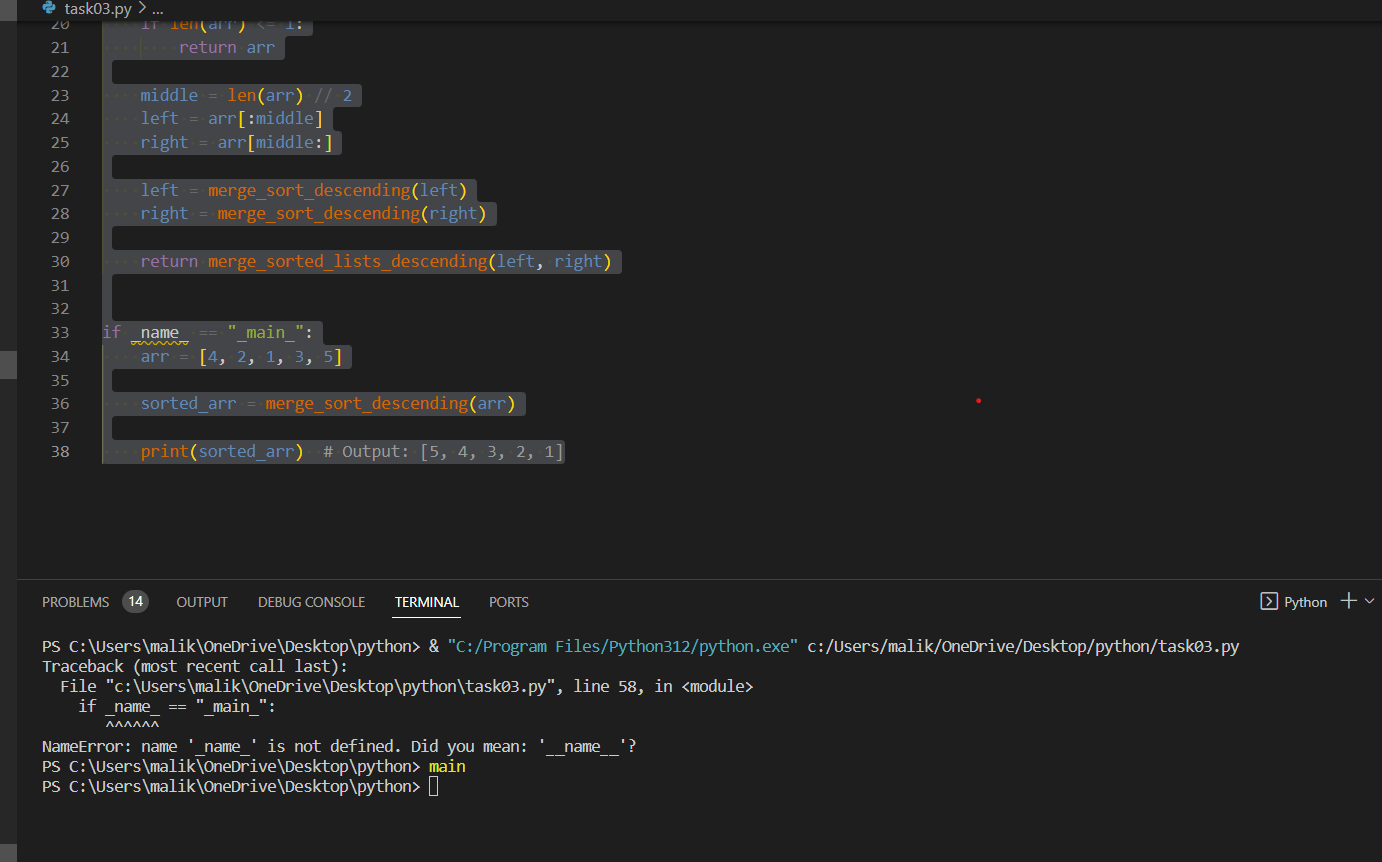
    return merge\_sorted\_lists\_descending(left, right)

if \_name\_ == "\_main\_":

    arr = [4, 2, 1, 3, 5]

    sorted\_arr = merge\_sort\_descending(arr)

    print(sorted\_arr)  # Output: [5, 4, 3, 2, 1]



Question 04:

def merge\_sorted\_lists\_descending(arrs):

    merged = []

    while any(arrs):

        max\_values = [float('-inf')] \* len(arrs)

        max\_indices = []

        for i in range(len(arrs)):

            if arrs[i] and arrs[i][-1] >= max\_values[i]:

                max\_values[i] = arrs[i][-1]

        for i in range(len(arrs)):

            if arrs[i] and arrs[i][-1] == max\_values[i]:

                max\_indices.append(i)

        for idx in max\_indices:

            merged.append(arrs[idx].pop())

    return merged

def merge\_sort\_descending(arr):

    if len(arr) <= 1:

        return arr

    sublists = [arr[i:i+1] for i in range(0, len(arr), 1)]  # Split the list into single-element sublists

    while len(sublists) > 1:

        merged\_sublists = []

        for i in range(0, len(sublists), 3):

            if i + 2 < len(sublists):

                merged\_sublists.append(merge\_sorted\_lists\_descending(sublists[i:i+3]))

            else:

                merged\_sublists.append(merge\_sorted\_lists\_descending(sublists[i:]))

        sublists = merged\_sublists

    return sublists[0]

if \_name\_ == "\_main\_":

    arr = [4, 2, 1, 3, 5, 9, 7, 6]

    sorted\_arr = merge\_sort\_descending(arr)

    print(sorted\_arr)  # Output: [9, 7, 6, 5, 4, 3, 2, 1]

