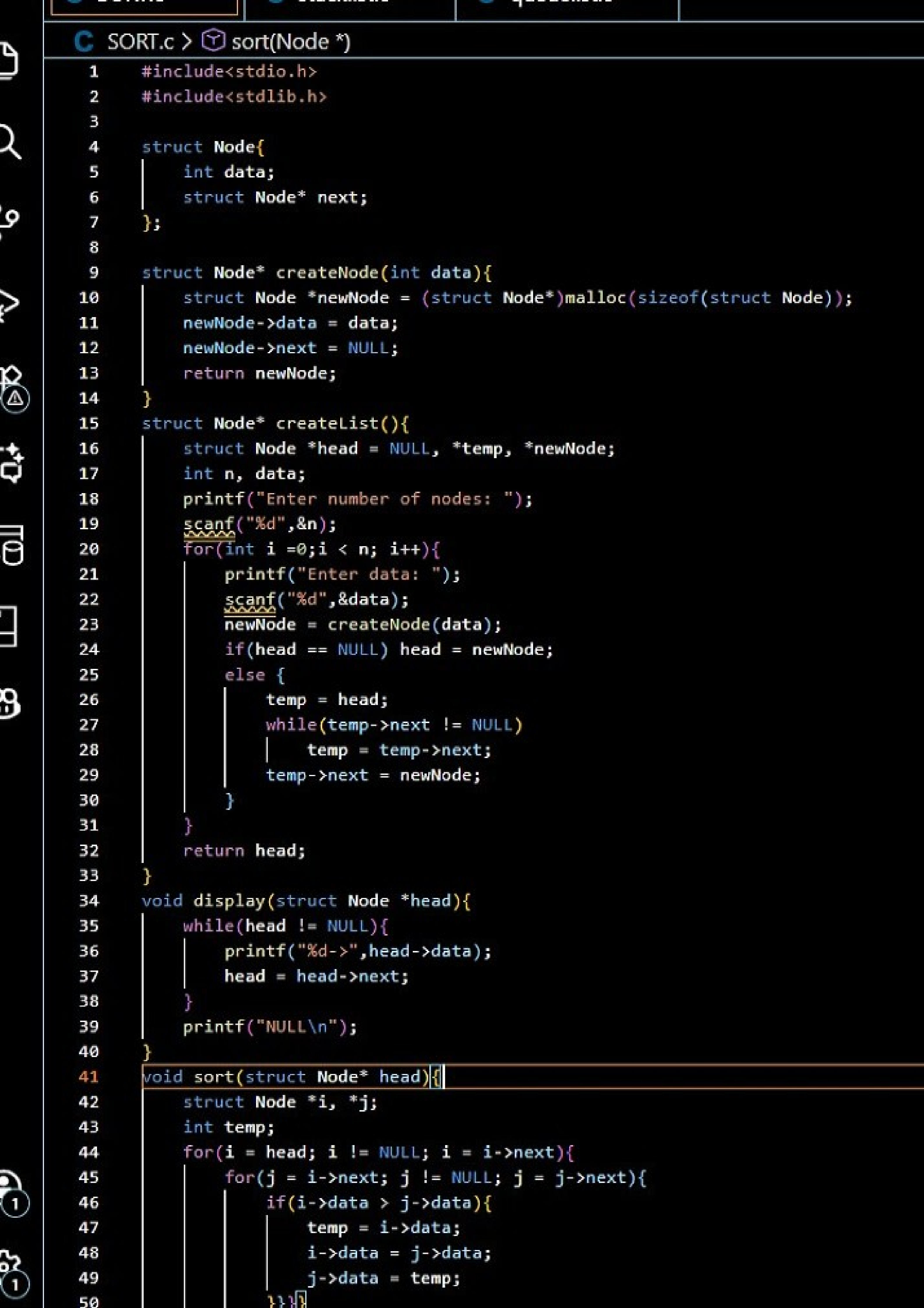
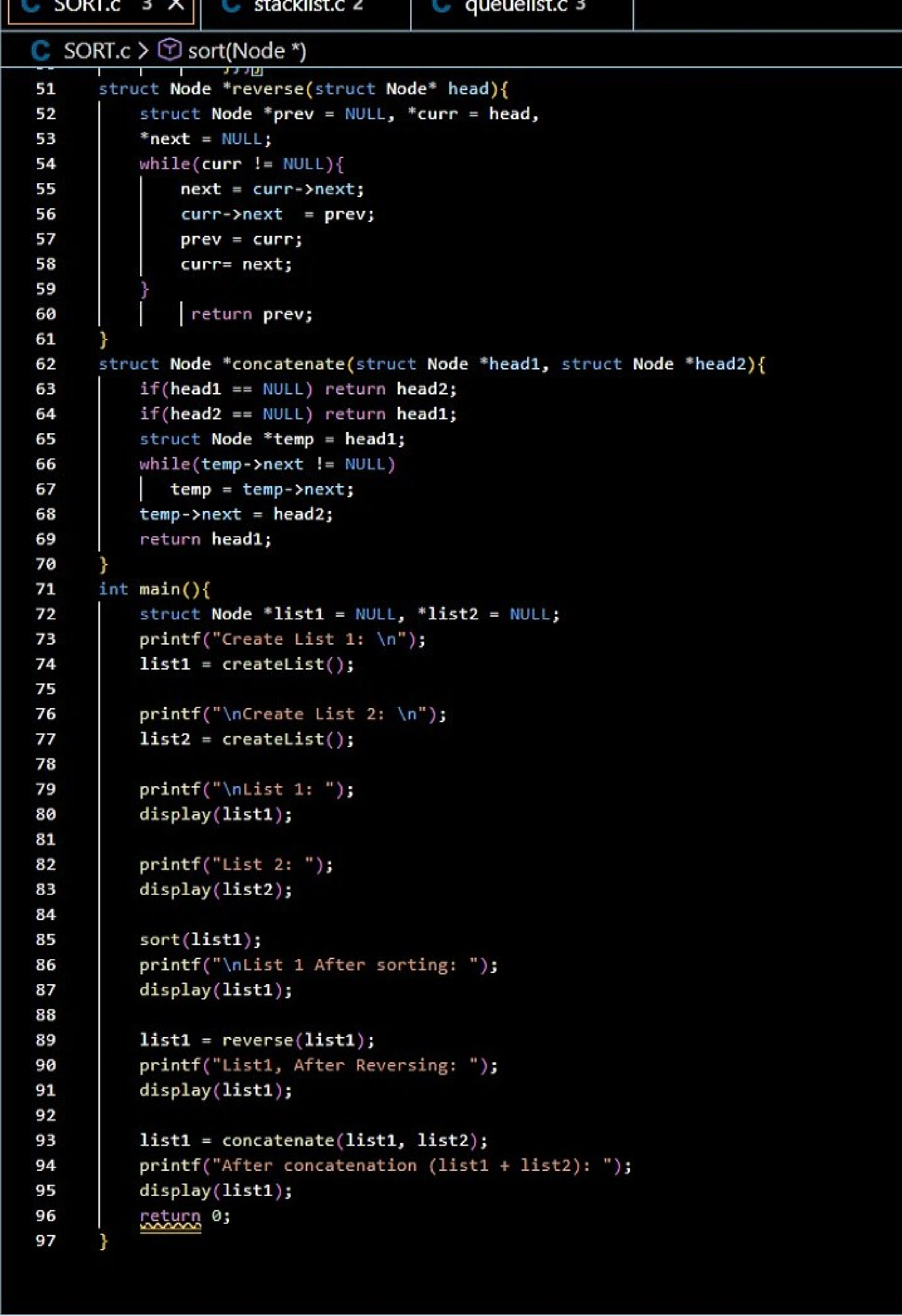
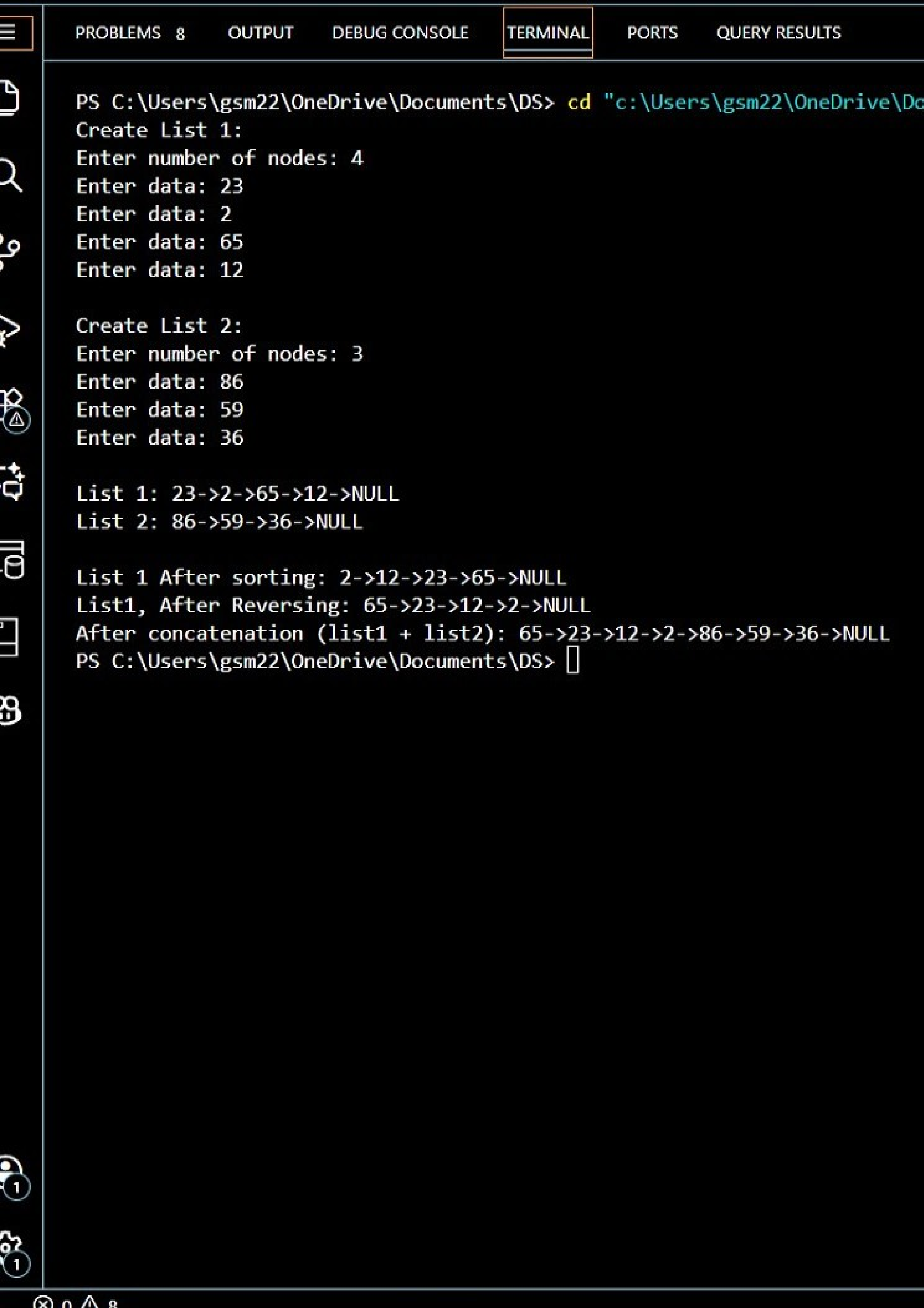


```c
#include<stdio.h>
#include<stdlib.h>

struct Node{
    int data;
    struct Node* next;
};

struct Node* createNode(int data){
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
struct Node* createList(){
    struct Node *head = NULL, *temp, *newNode;
    int n, data;
    printf("Enter number of nodes: ");
    scanf("%d",&n);
    for(int i =0;i < n; i++){
        printf("Enter data: ");
        scanf("%d",&data);
        newNode = createNode(data);
        if(head == NULL) head = newNode;
        else {
            temp = head;
            while(temp->next != NULL)
                temp = temp->next;
            temp->next = newNode;
        }
    }
    return head;
}
void display(struct Node *head){
    while(head != NULL){
        printf("%d->",head->data);
        head = head->next;
    }
    printf("NULL\n");
}
void sort(struct Node* head){
    struct Node *i, *j;
    int temp;
    for(i = head; i != NULL; i = i->next){
        for(j = i->next; j != NULL; j = j->next){
            if(i->data > j->data){
                temp = i->data;
                i->data = j->data;
                j->data = temp;
            }}}}
```



```
struct Node *reverse(struct Node* head){
    struct Node *prev = NULL, *curr = head,
    *next = NULL;
    while(curr != NULL){
        next = curr->next;
        curr->next  = prev;
        prev = curr;
        curr= next;
    }
         return prev;
}
struct Node *concatenate(struct Node *head1, struct Node *head2){
    if(head1 == NULL) return head2;
    if(head2 == NULL) return head1;
    struct Node *temp = head1;
    while(temp->next != NULL)
       temp = temp->next;
    temp->next = head2;
    return head1;
}
int main(){
    struct Node *list1 = NULL, *list2 = NULL;
    printf("Create List 1: \n");
    list1 = createList();

    printf("\nCreate List 2: \n");
    list2 = createList();

    printf("\nList 1: ");
    display(list1);

    printf("List 2: ");
    display(list2);

    sort(list1);
    printf("\nList 1 After sorting: ");
    display(list1);

    list1 = reverse(list1);
    printf("List1, After Reversing: ");
    display(list1);

    list1 = concatenate(list1, list2);
    printf("After concatenation (list1 + list2): ");
    display(list1);
    return 0;
}
```



```
PS C:\Users\gsm22\OneDrive\Documents\DS> cd "c:\Users\gsm22\OneDrive\Do
Create List 1:
Enter number of nodes: 4
Enter data: 23
Enter data: 2
Enter data: 65
Enter data: 12

Create List 2:
Enter number of nodes: 3
Enter data: 86
Enter data: 59
Enter data: 36

List 1: 23->2->65->12->NULL
List 2: 86->59->36->NULL

List 1 After sorting: 2->12->23->65->NULL
List1, After Reversing: 65->23->12->2->NULL
After concatenation (list1 + list2): 65->23->12->2->86->59->36->NULL
PS C:\Users\gsm22\OneDrive\Documents\DS> 
```