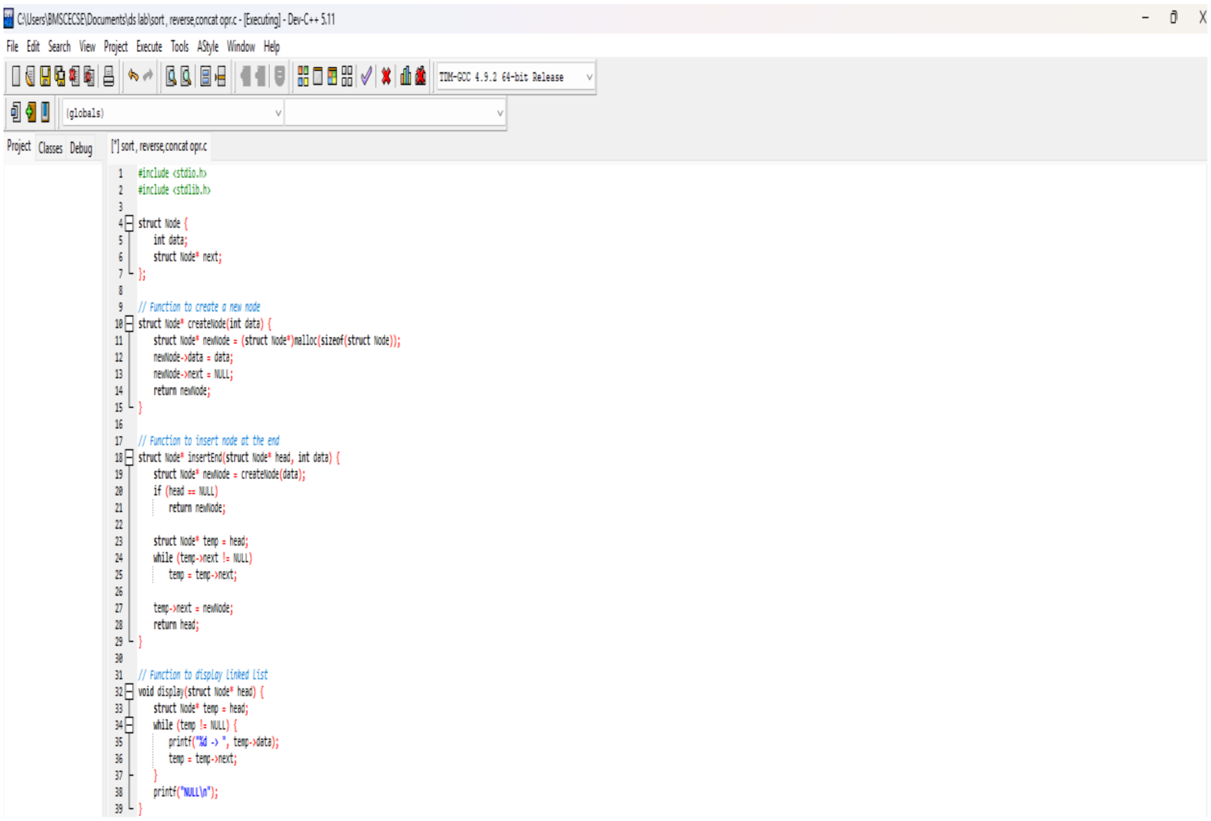
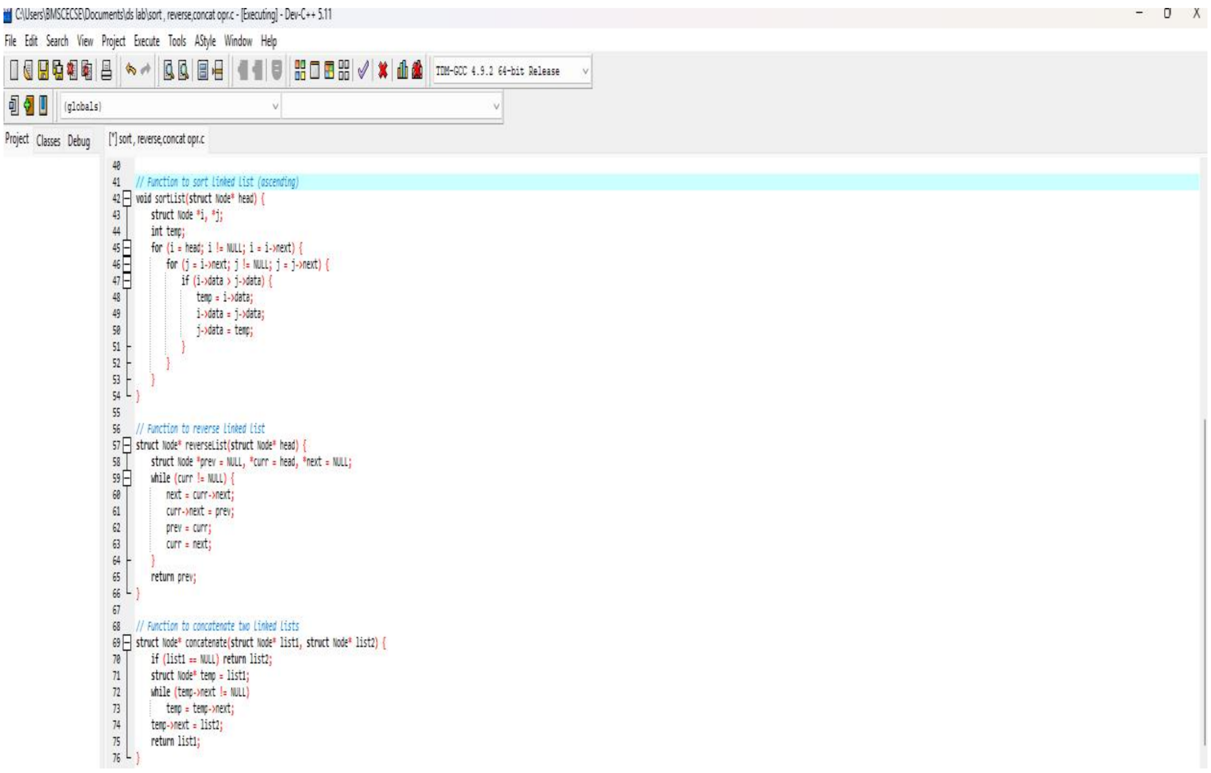
Program 6a



C:\Users\BMSCECSE\Documents\ds lab\sort , reverse,concat opr.c - [Executing] - Dev-C++ 5.11

[*] sort , reverse,concat opr.c

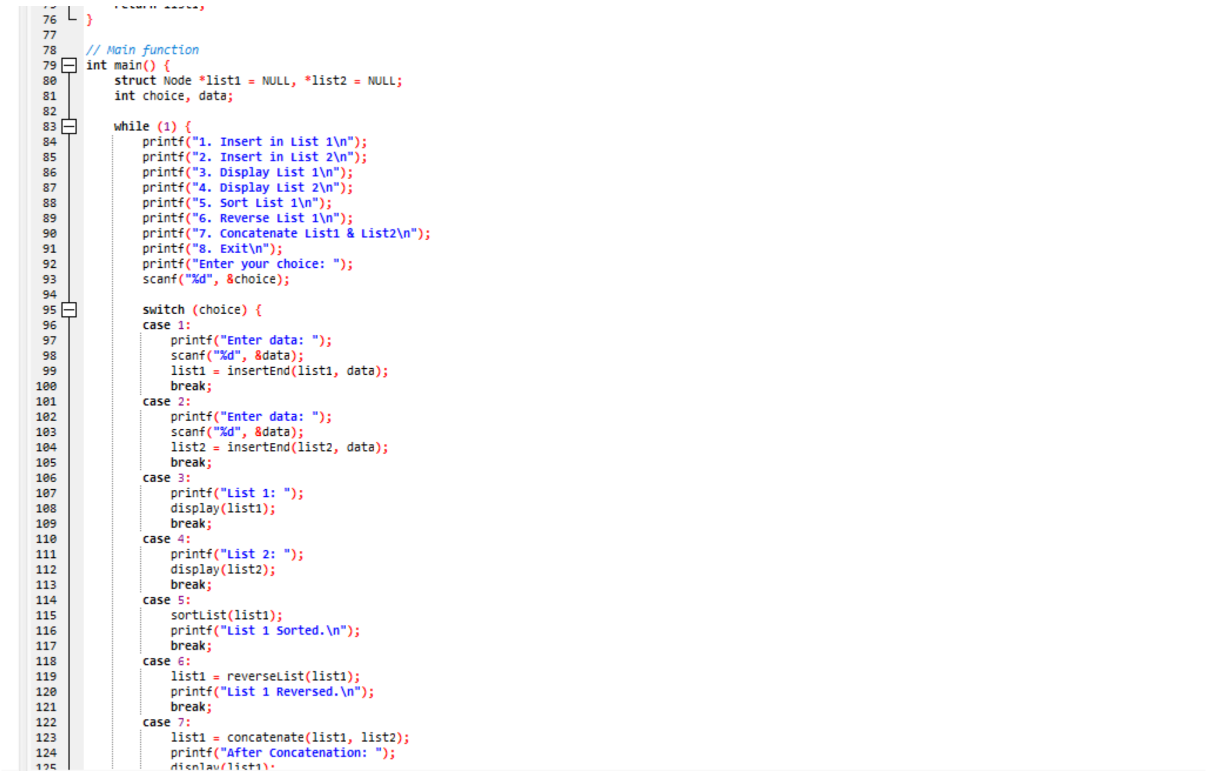
```c
#include <stdio.h>
#include <stdlib.h>

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

// Function to create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to insert node at the end
struct Node* insertEnd(struct Node* head, int data) {
    struct Node* newNode = createNode(data);
    if (head == NULL)
        return newNode;

    struct Node* temp = head;
    while (temp->next != NULL)
        temp = temp->next;

    temp->next = newNode;
    return head;
}

// Function to display linked list
void display(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
```



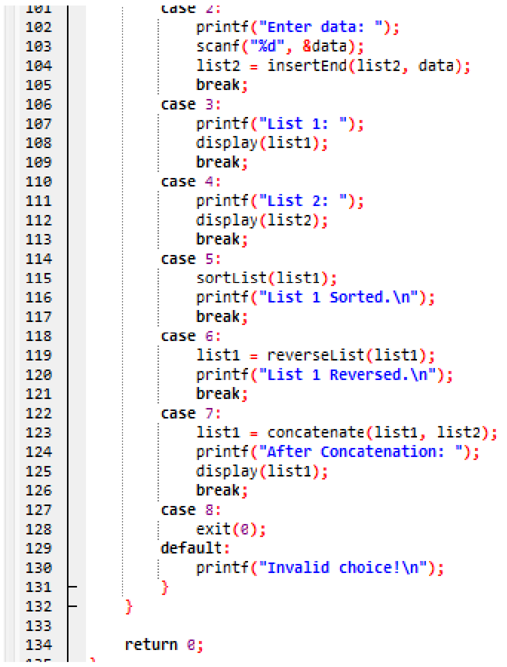
C:\Users\BMSCECSE\Documents\ds lab\sort , reverse,concat opr.c - [Executing] - Dev-C++ 5.11

[*] sort , reverse,concat opr.c

```c

// Function to sort linked list (ascending)
void sortList(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;
            }
        }
    }
}

// Function to reverse linked list
struct Node* reverseList(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;
}

// Function to concatenate two linked lists
struct Node* concatenate(struct Node* list1, struct Node* list2) {
    if (list1 == NULL) return list2;
    struct Node* temp = list1;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = list2;
    return list1;
}
```

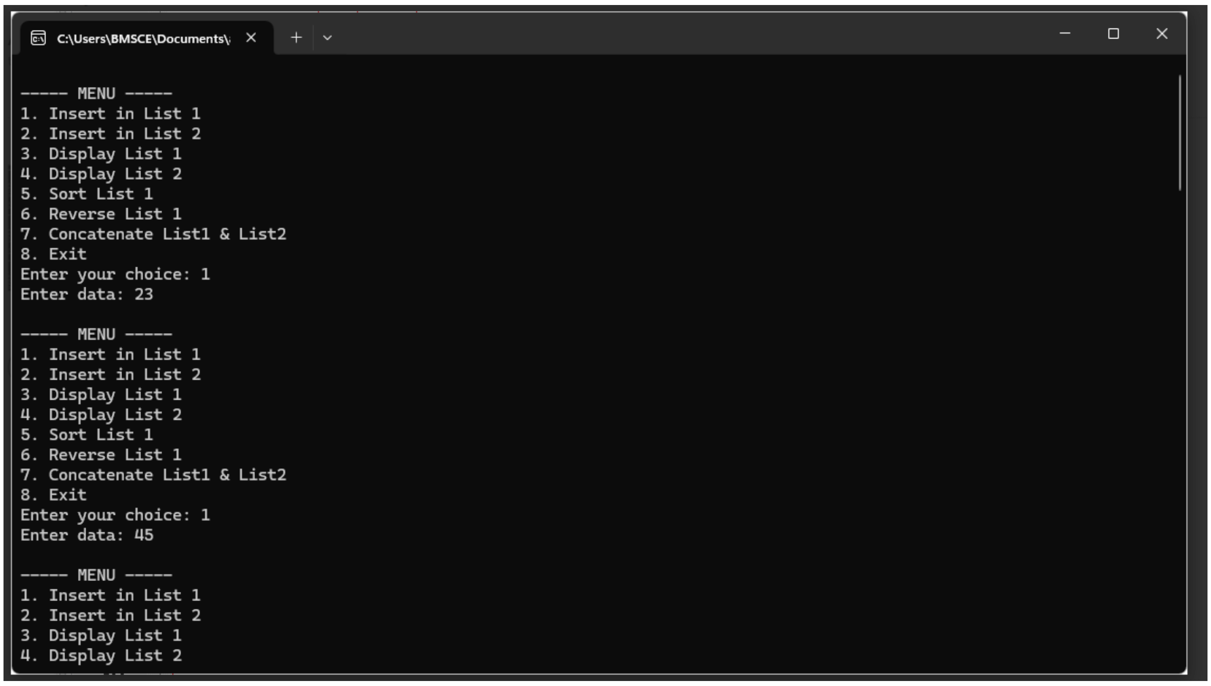


```c
    return list1;
}

// Main function
int main() {
    struct Node *list1 = NULL, *list2 = NULL;
    int choice, data;

    while (1) {
        printf("1. Insert in List 1\n");
        printf("2. Insert in List 2\n");
        printf("3. Display List 1\n");
        printf("4. Display List 2\n");
        printf("5. Sort List 1\n");
        printf("6. Reverse List 1\n");
        printf("7. Concatenate List1 & List2\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            printf("Enter data: ");
            scanf("%d", &data);
            list1 = insertEnd(list1, data);
            break;
        case 2:
            printf("Enter data: ");
            scanf("%d", &data);
            list2 = insertEnd(list2, data);
            break;
        case 3:
            printf("List 1: ");
            display(list1);
            break;
        case 4:
            printf("List 2: ");
            display(list2);
            break;
        case 5:
            sortList(list1);
            printf("List 1 Sorted.\n");
            break;
        case 6:
            list1 = reverseList(list1);
            printf("List 1 Reversed.\n");
            break;
        case 7:
            list1 = concatenate(list1, list2);
            printf("After Concatenation: ");
            display(list1);
```

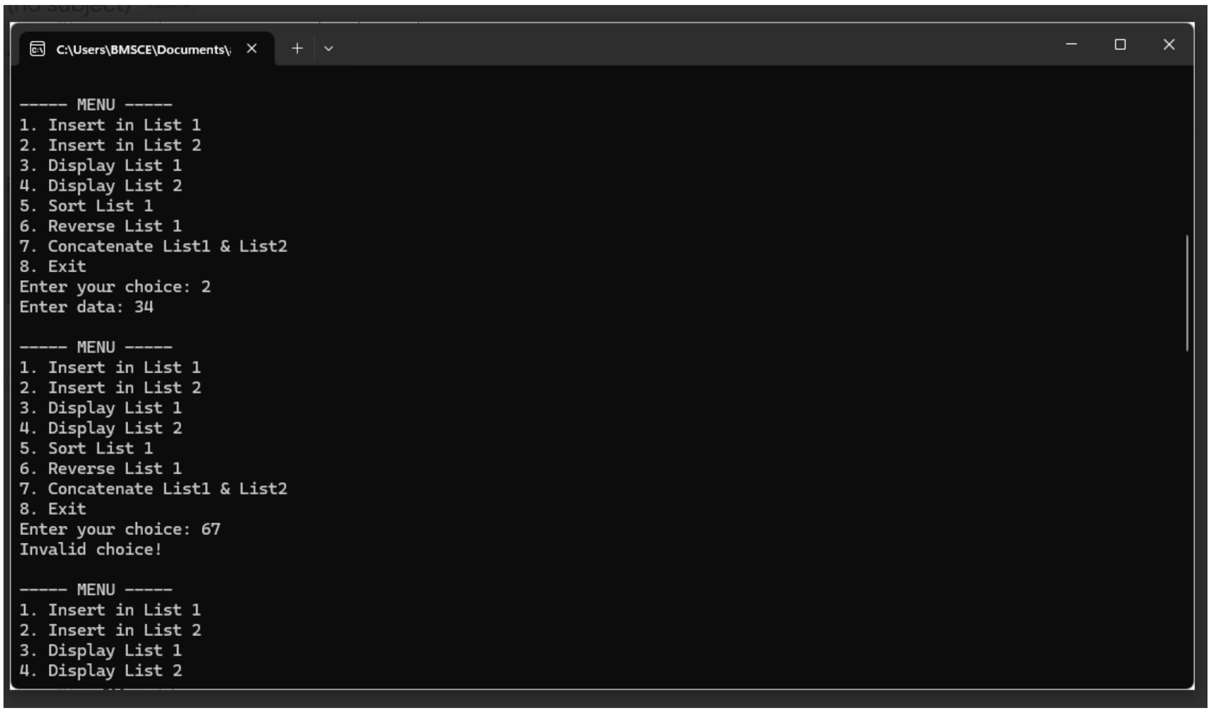


```c
        case 2:
            printf("Enter data: ");
            scanf("%d", &data);
            list2 = insertEnd(list2, data);
            break;
        case 3:
            printf("List 1: ");
            display(list1);
            break;
        case 4:
            printf("List 2: ");
            display(list2);
            break;
        case 5:
            sortList(list1);
            printf("List 1 Sorted.\n");
            break;
        case 6:
            list1 = reverseList(list1);
            printf("List 1 Reversed.\n");
            break;
        case 7:
            list1 = concatenate(list1, list2);
            printf("After Concatenation: ");
            display(list1);
            break;
        case 8:
            exit(0);
        default:
            printf("Invalid choice!\n");
        }
    }

    return 0;
```

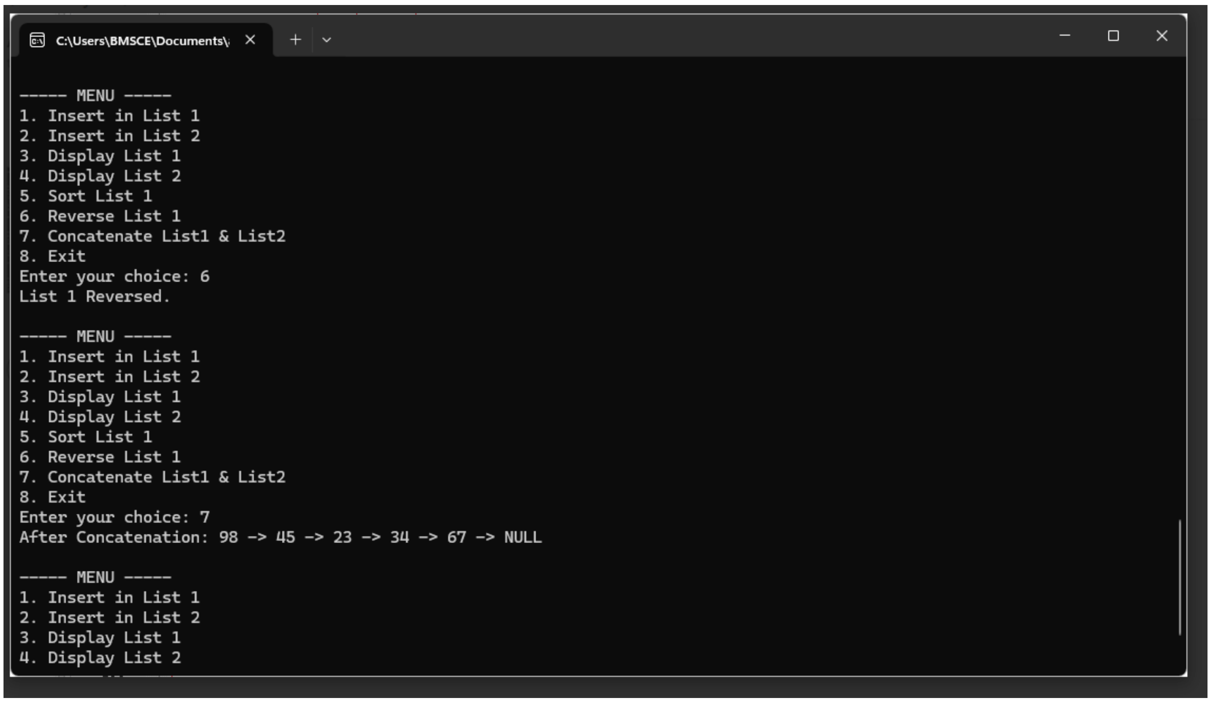
Output:



```
----- MENU -----
1. Insert in List 1
2. Insert in List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List1 & List2
8. Exit
Enter your choice: 1
Enter data: 23

----- MENU -----
1. Insert in List 1
2. Insert in List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List1 & List2
8. Exit
Enter your choice: 1
Enter data: 45

----- MENU -----
1. Insert in List 1
2. Insert in List 2
3. Display List 1
4. Display List 2
```



```
----- MENU -----
1. Insert in List 1
2. Insert in List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List1 & List2
8. Exit
Enter your choice: 2
Enter data: 34

----- MENU -----
1. Insert in List 1
2. Insert in List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List1 & List2
8. Exit
Enter your choice: 67
Invalid choice!

----- MENU -----
1. Insert in List 1
2. Insert in List 2
3. Display List 1
4. Display List 2
```



```
----- MENU -----
1. Insert in List 1
2. Insert in List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List1 & List2
8. Exit
Enter your choice: 6
List 1 Reversed.

----- MENU -----
1. Insert in List 1
2. Insert in List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List1 & List2
8. Exit
Enter your choice: 7
After Concatenation: 98 -> 45 -> 23 -> 34 -> 67 -> NULL

----- MENU -----
1. Insert in List 1
2. Insert in List 2
3. Display List 1
4. Display List 2
```