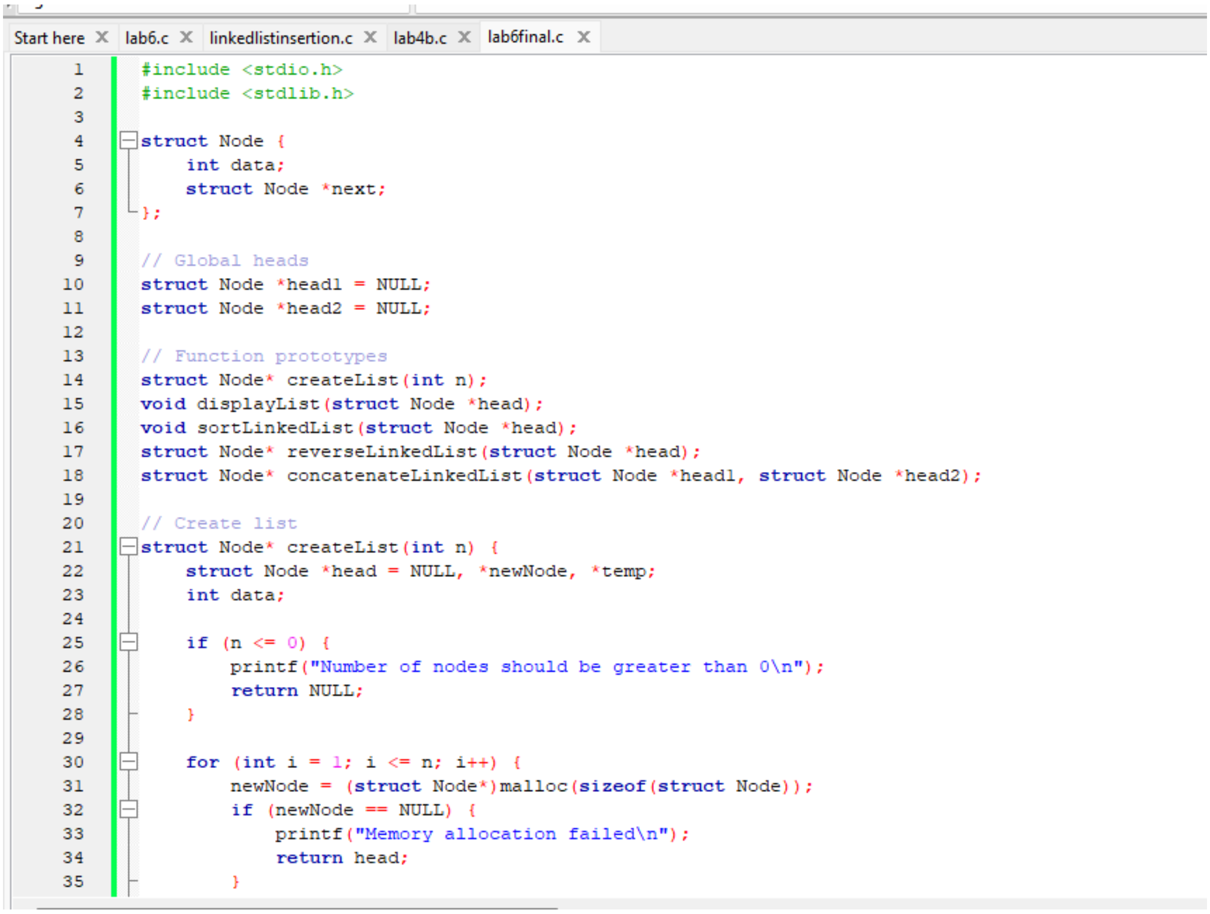
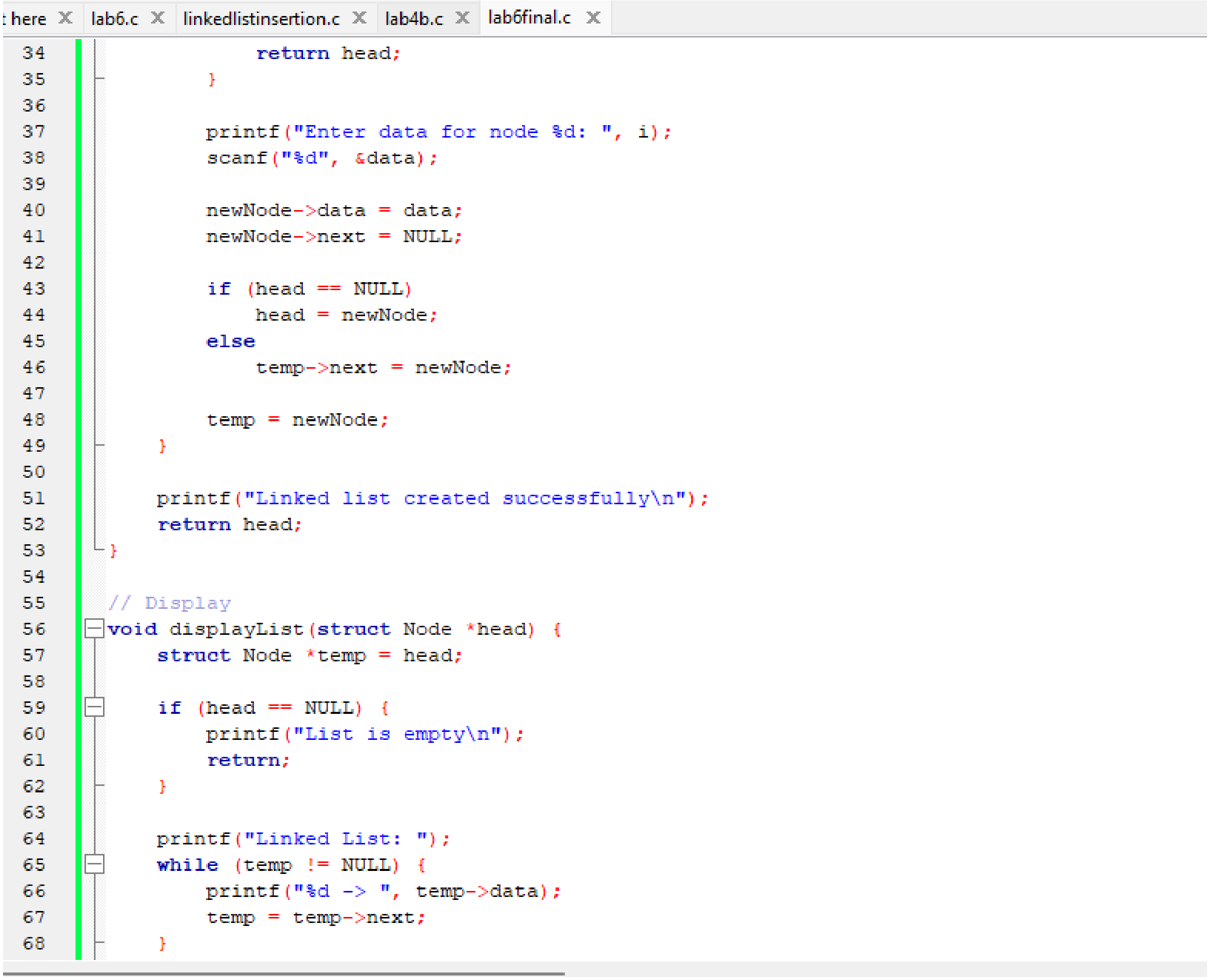
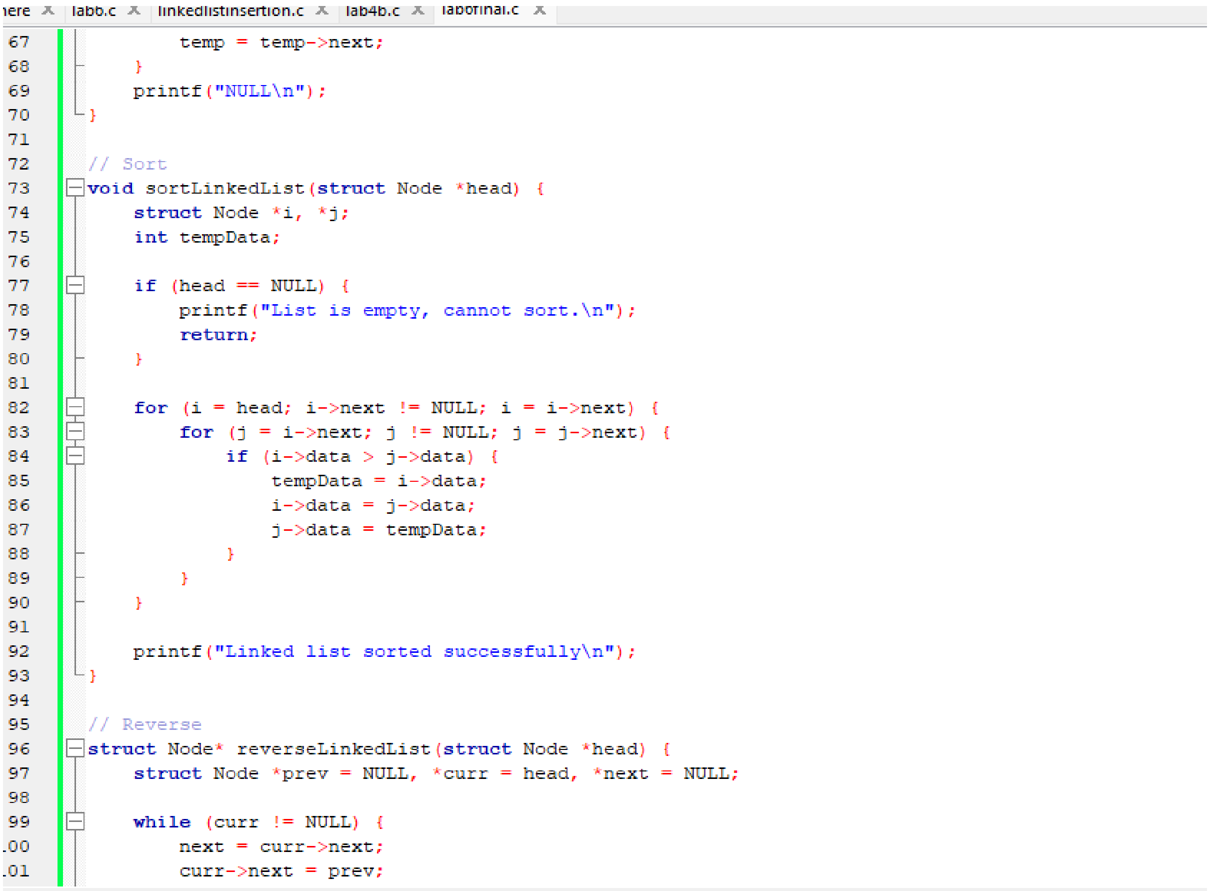


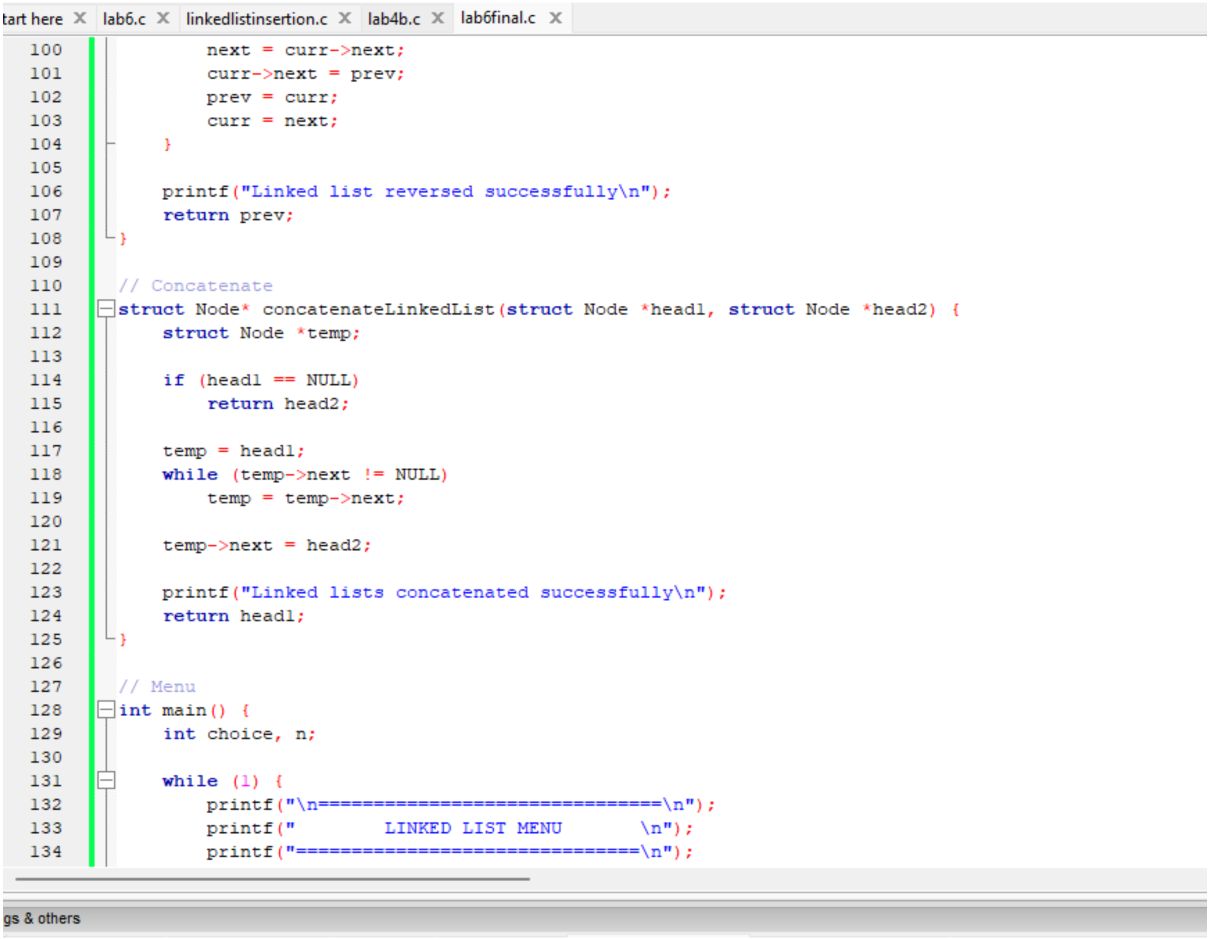
```c
#include <stdio.h>
#include <stdlib.h>

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

// Global heads
struct Node *head1 = NULL;
struct Node *head2 = NULL;

// Function prototypes
struct Node* createList(int n);
void displayList(struct Node *head);
void sortLinkedList(struct Node *head);
struct Node* reverseLinkedList(struct Node *head);
struct Node* concatenateLinkedList(struct Node *head1, struct Node *head2);

// Create list
struct Node* createList(int n) {
    struct Node *head = NULL, *newNode, *temp;
    int data;

    if (n <= 0) {
        printf("Number of nodes should be greater than 0\n");
        return NULL;
    }

    for (int i = 1; i <= n; i++) {
        newNode = (struct Node*)malloc(sizeof(struct Node));
        if (newNode == NULL) {
            printf("Memory allocation failed\n");
            return head;
        }
```



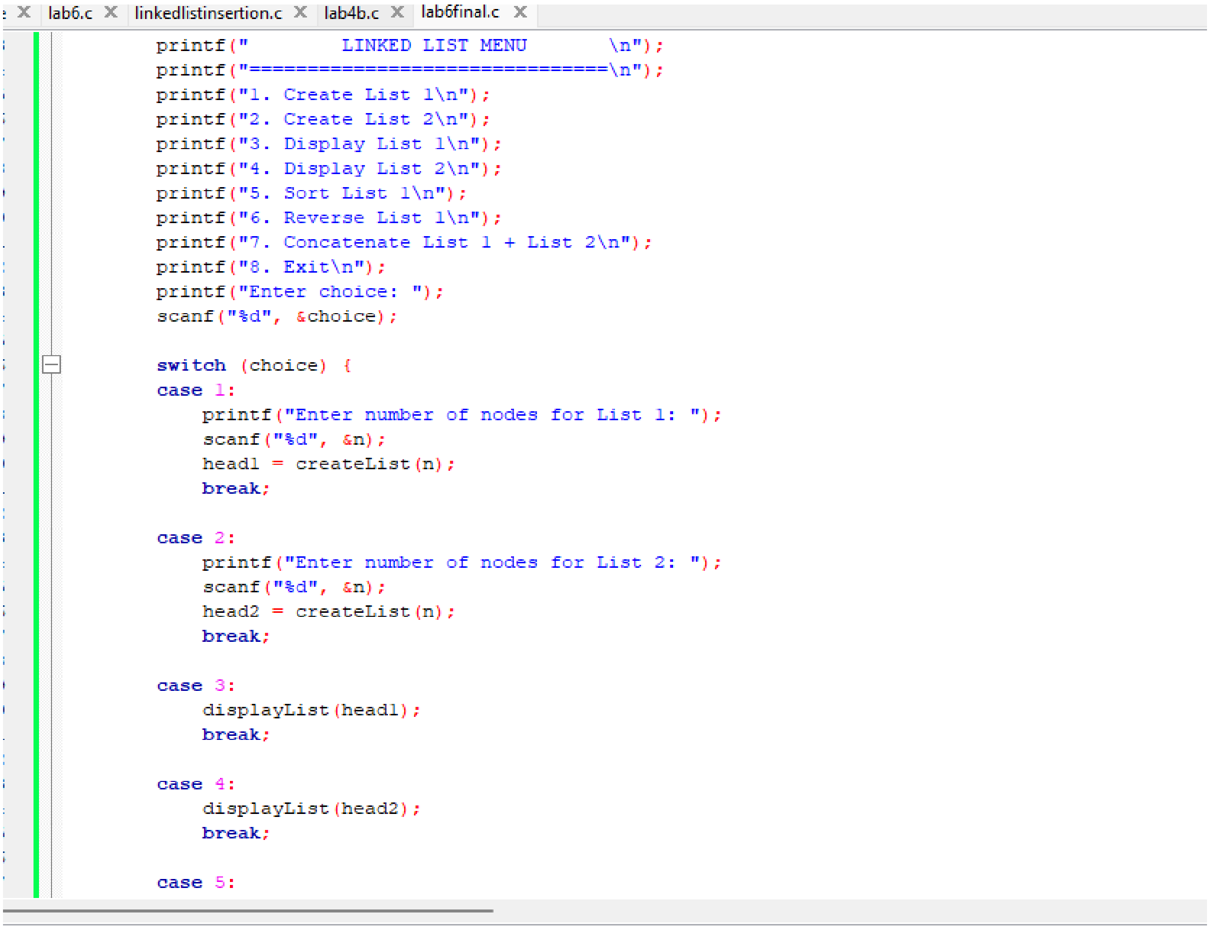
```c
            return head;
        }

        printf("Enter data for node %d: ", i);
        scanf("%d", &data);

        newNode->data = data;
        newNode->next = NULL;

        if (head == NULL)
            head = newNode;
        else
            temp->next = newNode;

        temp = newNode;
    }

    printf("Linked list created successfully\n");
    return head;
}

// Display
void displayList(struct Node *head) {
    struct Node *temp = head;

    if (head == NULL) {
        printf("List is empty\n");
        return;
    }

    printf("Linked List: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
```



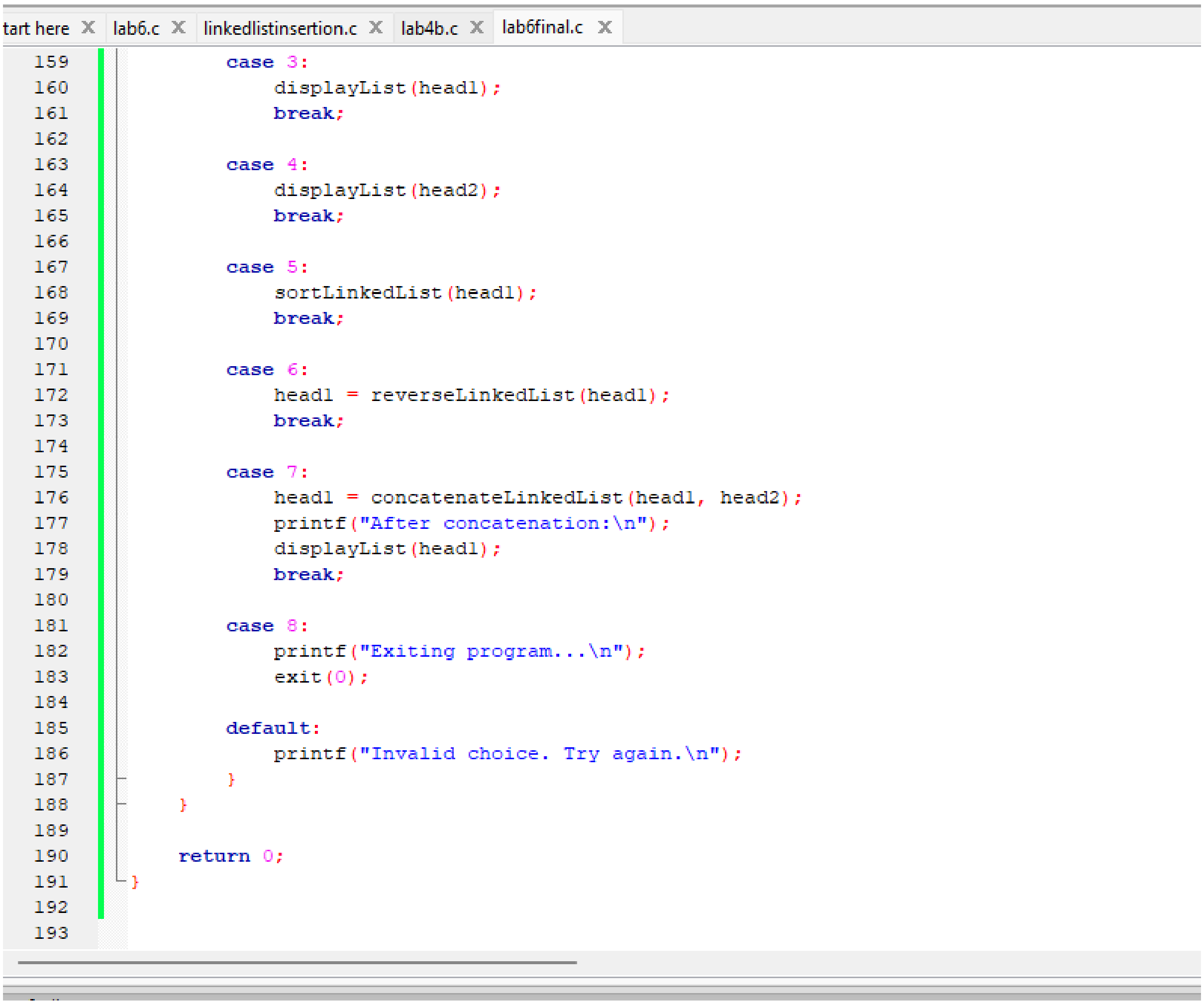
```c
            temp = temp->next;
        }
        printf("NULL\n");
}

// Sort
void sortLinkedList(struct Node *head) {
    struct Node *i, *j;
    int tempData;

    if (head == NULL) {
        printf("List is empty, cannot sort.\n");
        return;
    }

    for (i = head; i->next != NULL; i = i->next) {
        for (j = i->next; j != NULL; j = j->next) {
            if (i->data > j->data) {
                tempData = i->data;
                i->data = j->data;
                j->data = tempData;
            }
        }
    }

    printf("Linked list sorted successfully\n");
}

// Reverse
struct Node* reverseLinkedList(struct Node *head) {
    struct Node *prev = NULL, *curr = head, *next = NULL;

    while (curr != NULL) {
        next = curr->next;
        curr->next = prev;
```



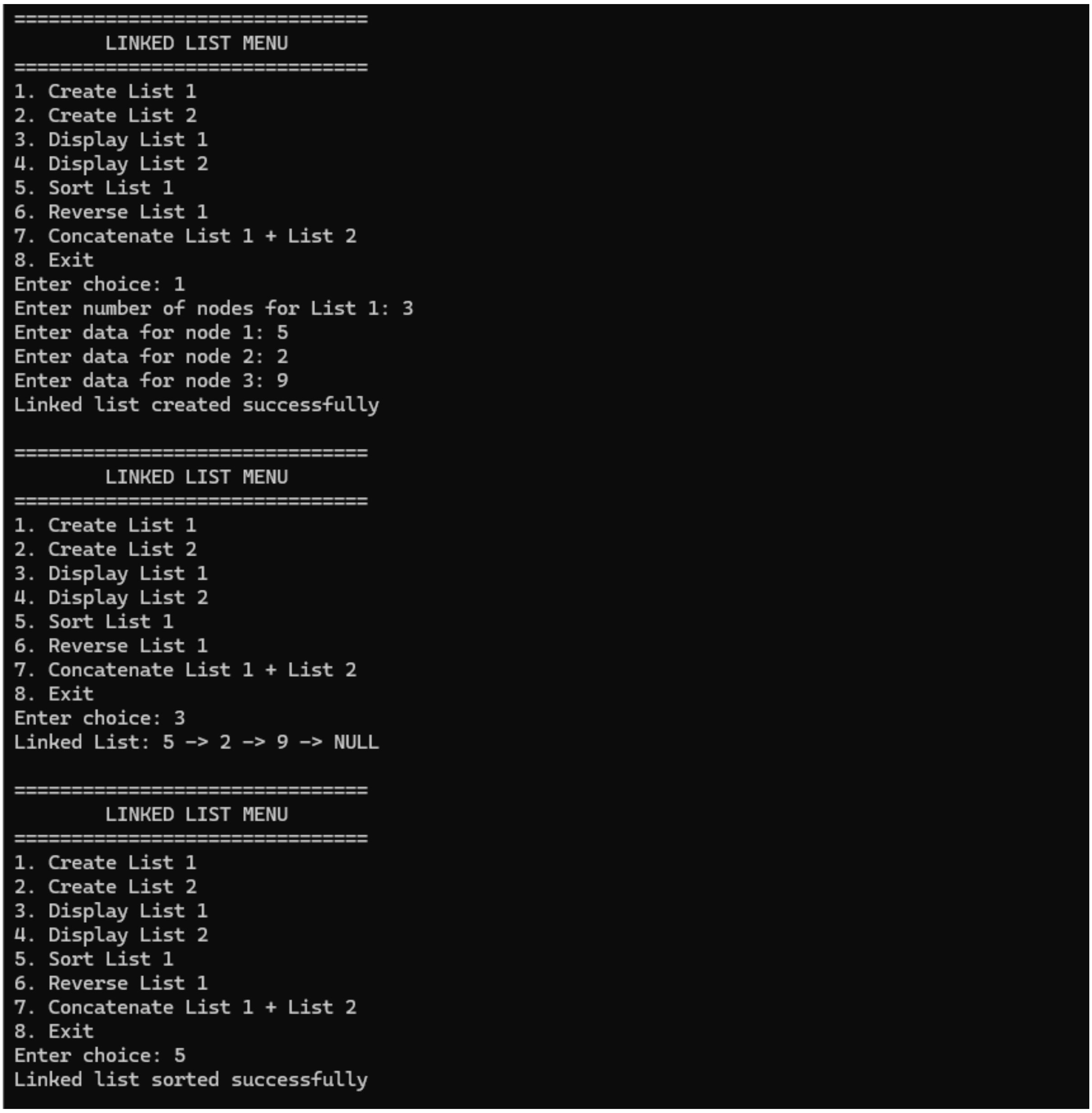
```c
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }

    printf("Linked list reversed successfully\n");
    return prev;
}

// Concatenate
struct Node* concatenateLinkedList(struct Node *head1, struct Node *head2) {
    struct Node *temp;

    if (head1 == NULL)
        return head2;

    temp = head1;
    while (temp->next != NULL)
        temp = temp->next;

    temp->next = head2;

    printf("Linked lists concatenated successfully\n");
    return head1;
}

// Menu
int main() {
    int choice, n;

    while (1) {
        printf("\n==============================\n");
        printf("        LINKED LIST MENU       \n");
        printf("==============================\n");
```



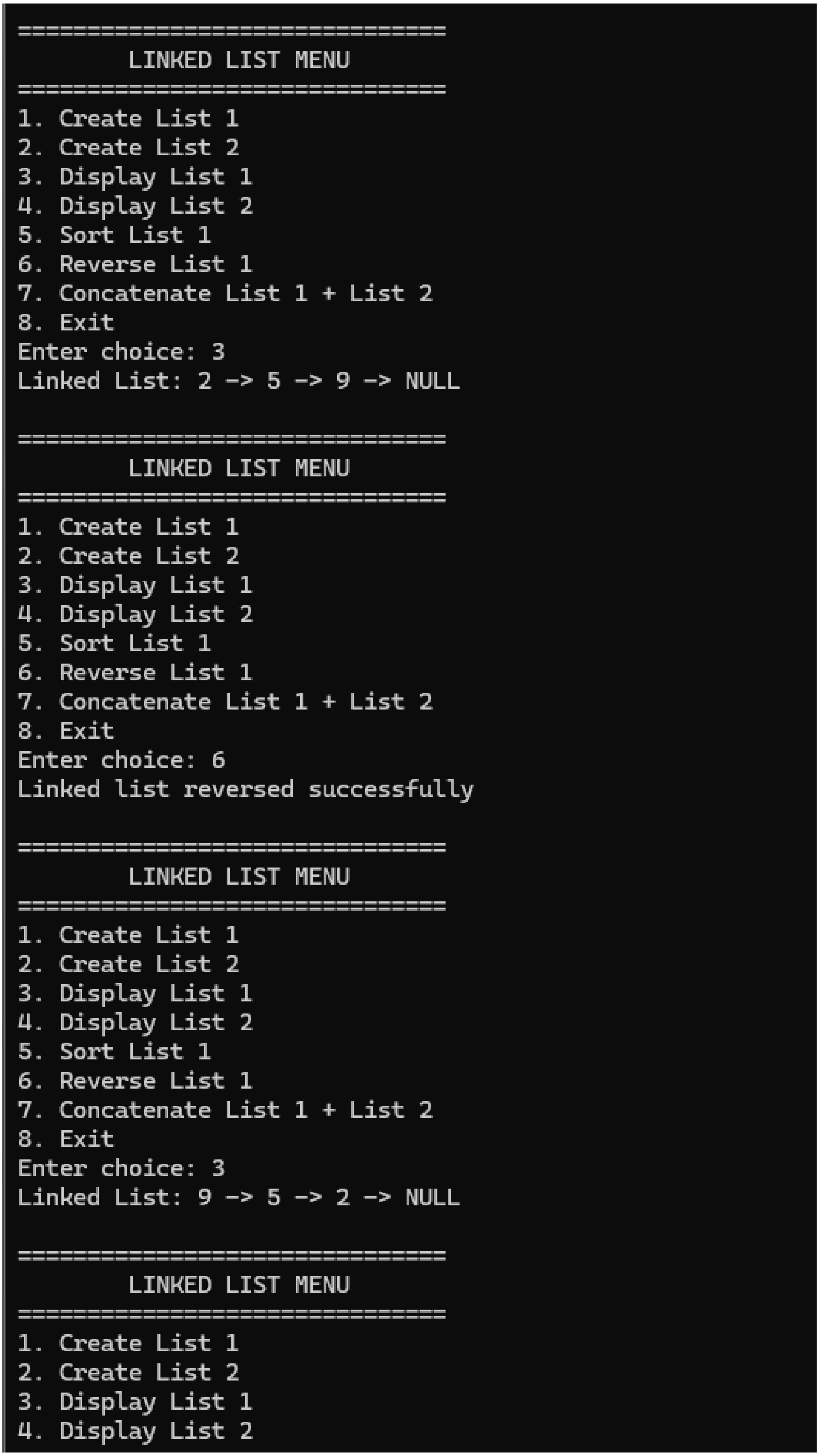
```
        printf("        LINKED LIST MENU       \n");
        printf("===============================\n");
        printf("1. Create List 1\n");
        printf("2. Create List 2\n");
        printf("3. Display List 1\n");
        printf("4. Display List 2\n");
        printf("5. Sort List 1\n");
        printf("6. Reverse List 1\n");
        printf("7. Concatenate List 1 + List 2\n");
        printf("8. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            printf("Enter number of nodes for List 1: ");
            scanf("%d", &n);
            head1 = createList(n);
            break;

        case 2:
            printf("Enter number of nodes for List 2: ");
            scanf("%d", &n);
            head2 = createList(n);
            break;

        case 3:
            displayList(head1);
            break;

        case 4:
            displayList(head2);
            break;

        case 5:
```



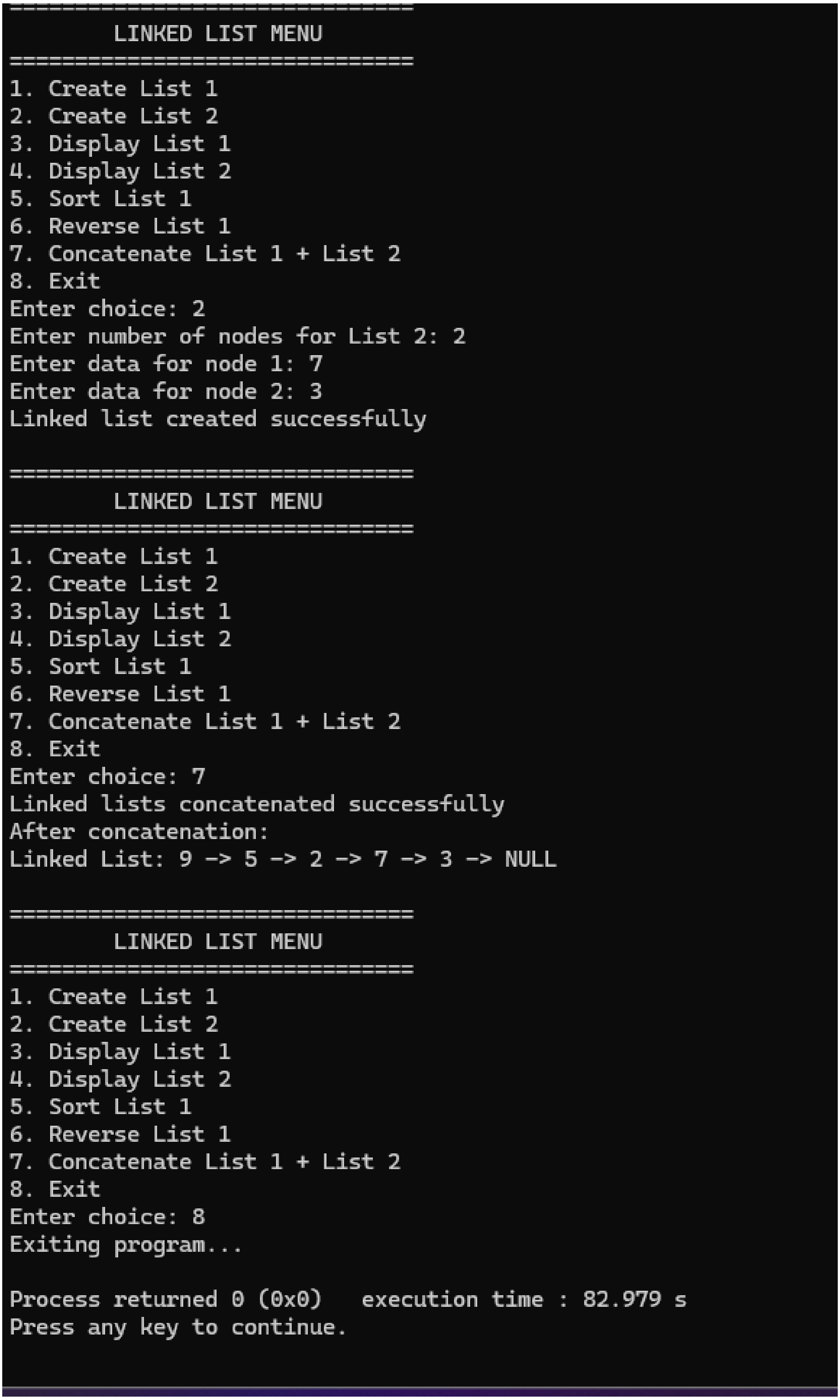
```c
        case 3:
            displayList(head1);
            break;

        case 4:
            displayList(head2);
            break;

        case 5:
            sortLinkedList(head1);
            break;

        case 6:
            head1 = reverseLinkedList(head1);
            break;

        case 7:
            head1 = concatenateLinkedList(head1, head2);
            printf("After concatenation:\n");
            displayList(head1);
            break;

        case 8:
            printf("Exiting program...\n");
            exit(0);

        default:
            printf("Invalid choice. Try again.\n");
        }
    }

    return 0;
}
```



```
===============================
        LINKED LIST MENU
===============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 1
Enter number of nodes for List 1: 3
Enter data for node 1: 5
Enter data for node 2: 2
Enter data for node 3: 9
Linked list created successfully

===============================
        LINKED LIST MENU
===============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 3
Linked List: 5 -> 2 -> 9 -> NULL

===============================
        LINKED LIST MENU
===============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 5
Linked list sorted successfully
```



```
===============================
        LINKED LIST MENU
===============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 3
Linked List: 2 -> 5 -> 9 -> NULL

===============================
        LINKED LIST MENU
===============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 6
Linked list reversed successfully

===============================
        LINKED LIST MENU
===============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 3
Linked List: 9 -> 5 -> 2 -> NULL

===============================
        LINKED LIST MENU
===============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
```



```
===============================
        LINKED LIST MENU
===============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 2
Enter number of nodes for List 2: 2
Enter data for node 1: 7
Enter data for node 2: 3
Linked list created successfully

===============================
        LINKED LIST MENU
===============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 7
Linked lists concatenated successfully
After concatenation:
Linked List: 9 -> 5 -> 2 -> 7 -> 3 -> NULL

===============================
        LINKED LIST MENU
===============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 8
Exiting program...

Process returned 0 (0x0)   execution time : 82.979 s
Press any key to continue.
```