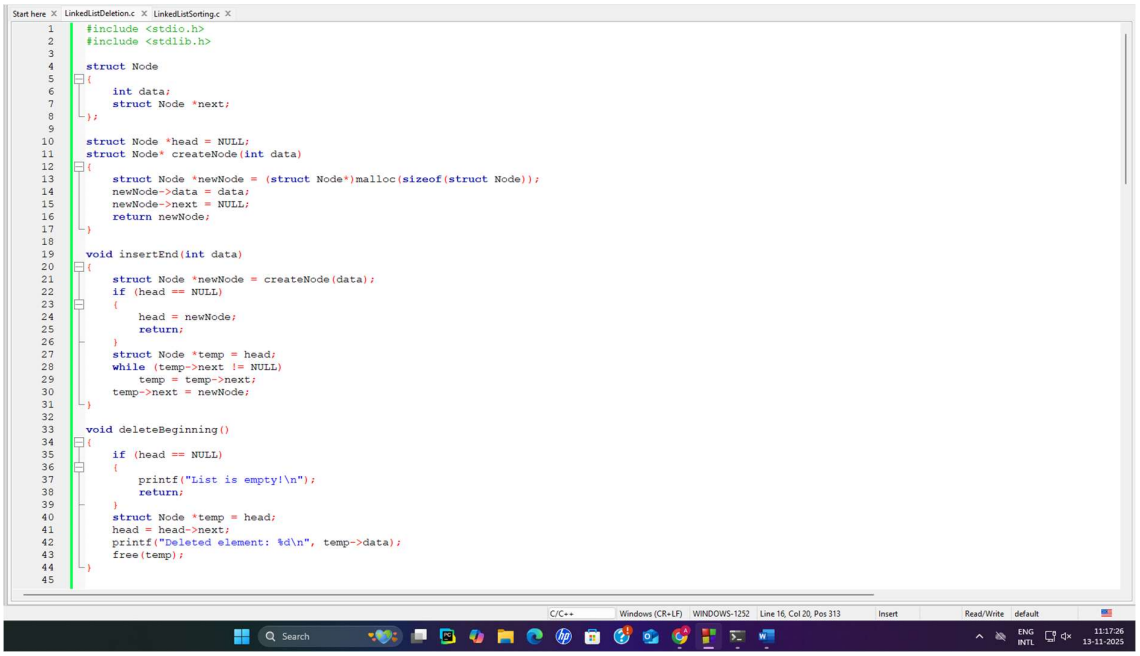
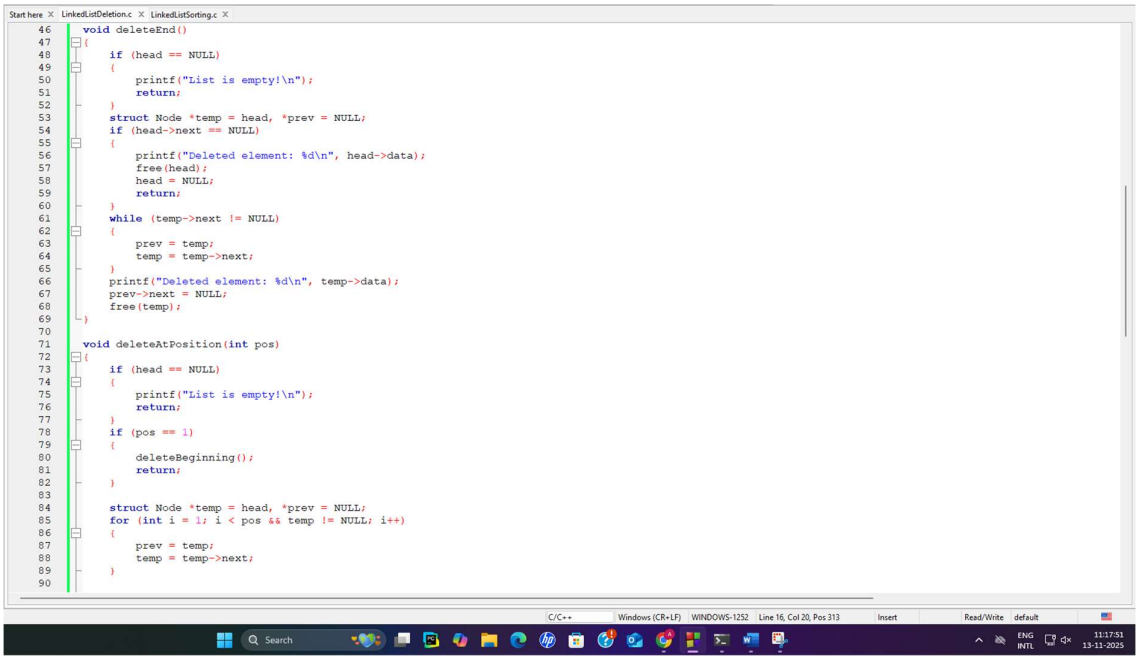
LAB PROGRAM - 6

a). SLL - Delete (start,end,any position)



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

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

struct Node *head = NULL;
struct Node* createNode(int data)
{
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void insertEnd(int data)
{
    struct Node *newNode = createNode(data);
    if (head == NULL)
    {
        head = newNode;
        return;
    }
    struct Node *temp = head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
}

void deleteBeginning()
{
    if (head == NULL)
    {
        printf("List is empty!\n");
        return;
    }
    struct Node *temp = head;
    head = head->next;
    printf("Deleted element: %d\n", temp->data);
    free(temp);
}
```



```c
void deleteEnd()
{
    if (head == NULL)
    {
        printf("List is empty!\n");
        return;
    }
    struct Node *temp = head, *prev = NULL;
    if (head->next == NULL)
    {
        printf("Deleted element: %d\n", head->data);
        free(head);
        head = NULL;
        return;
    }
    while (temp->next != NULL)
    {
        prev = temp;
        temp = temp->next;
    }
    printf("Deleted element: %d\n", temp->data);
    prev->next = NULL;
    free(temp);
}

void deleteAtPosition(int pos)
{
    if (head == NULL)
    {
        printf("List is empty!\n");
        return;
    }
    if (pos == 1)
    {
        deleteBeginning();
        return;
    }

    struct Node *temp = head, *prev = NULL;
    for (int i = 1; i < pos && temp != NULL; i++)
    {
        prev = temp;
        temp = temp->next;
    }
```

Start here | LinkedListDeletion.c | LinkedListSorting.c

```c
    }

    struct Node *temp = head, *prev = NULL;
    for (int i = 1; i < pos && temp != NULL; i++)
    {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL)
    {
        printf("Position not found!\n");
        return;
    }

    prev->next = temp->next;
    printf("Deleted element: %d\n", temp->data);
    free(temp);
}

void display()
{
    if (head == NULL)
    {
        printf("List is empty!\n");
        return;
    }
    struct Node *temp = head;
    printf("Linked List: ");
    while (temp != NULL)
    {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main()
{
    int choice, data, pos;
    while(1)
    {
        printf("\n--- Linked List Deletion Menu ---\n");
        printf("1. Insert End\n");
        printf("2. Delete Beginning\n");
```

C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 114, Col 27, Pos 2185 | Insert | Read/Write | default

Search | ENG INTL | 11:19:15 13-11-2025

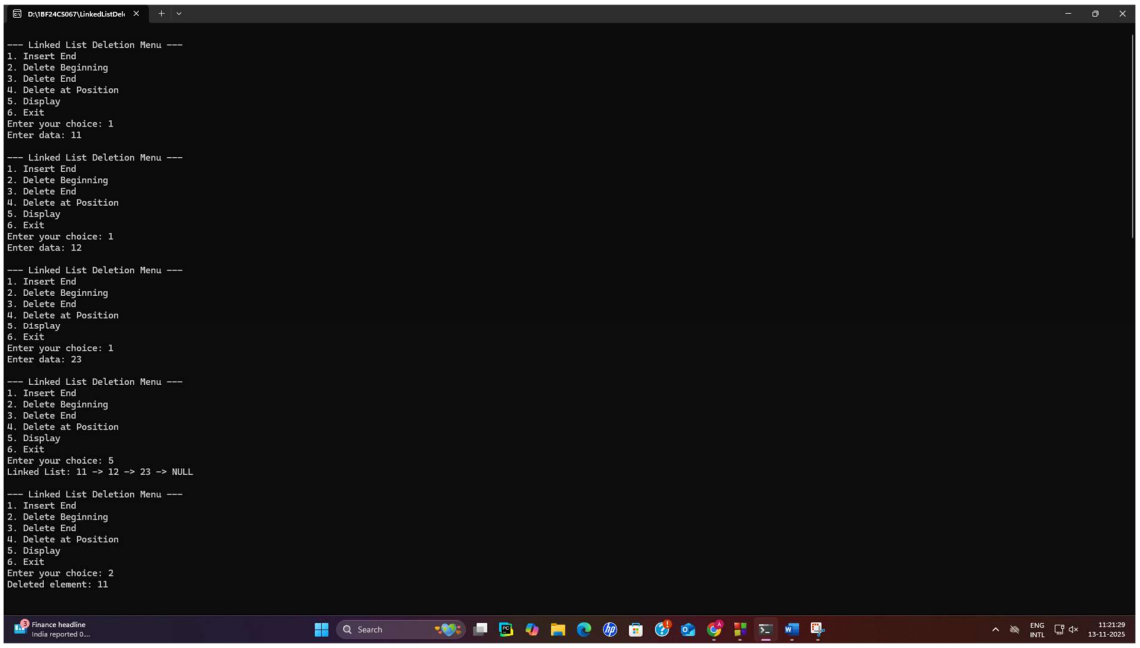
Start here | LinkedListDeletion.c | LinkedListSorting.c

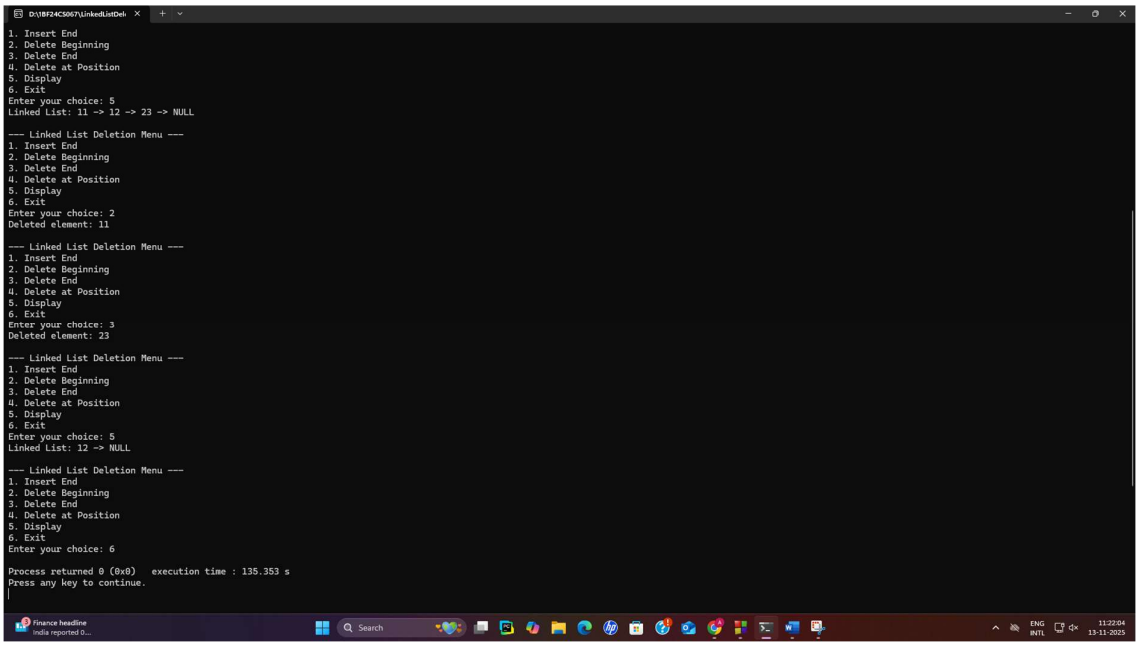
```c
int main()
{
    int choice, data, pos;
    while(1)
    {
        printf("\n--- Linked List Deletion Menu ---\n");
        printf("1. Insert End\n");
        printf("2. Delete Beginning\n");
        printf("3. Delete End\n");
        printf("4. Delete at Position\n");
        printf("5. Display\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                printf("Enter data: ");
                scanf("%d", &data);
                insertEnd(data);
                break;
            case 2:
                deleteBeginning();
                break;
            case 3:
                deleteEnd();
                break;
            case 4:
                printf("Enter position to delete: ");
                scanf("%d", &pos);
                deleteAtPosition(pos);
                break;
            case 5:
                display();
                break;
            case 6:
                exit(0);
            default:
                printf("Invalid choice!\n");
        }
    }
    return 0;
}
```

C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 114, Col 27, Pos 2185 | Insert | Read/Write | default

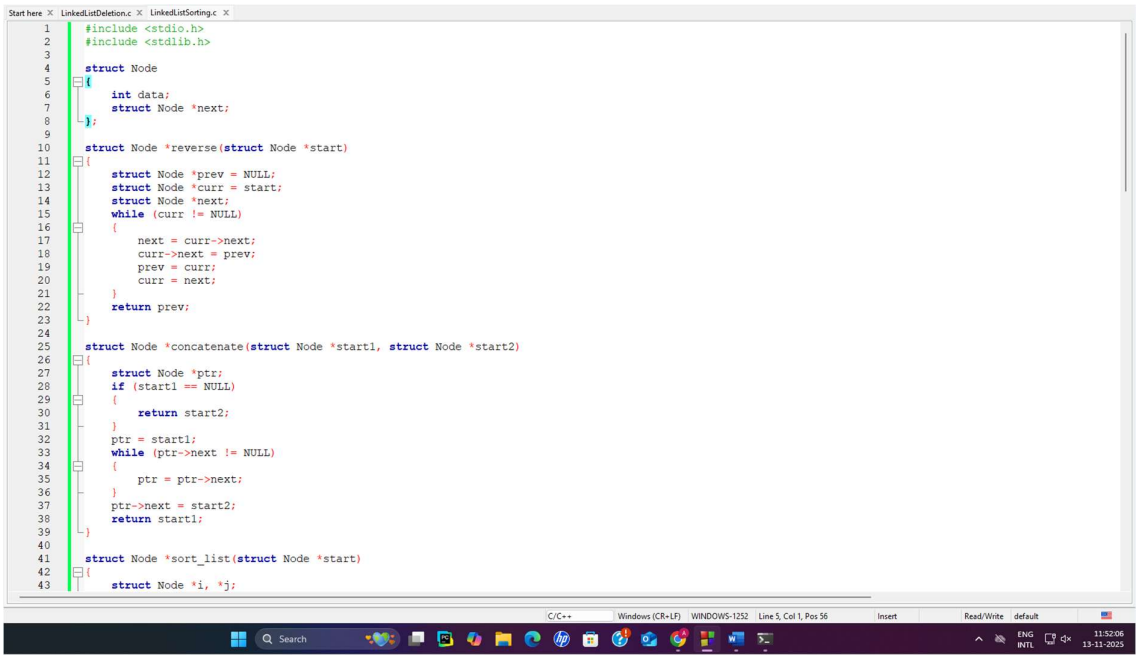
Search | ENG INTL | 11:19:38 13-11-2025

OUTPUT:

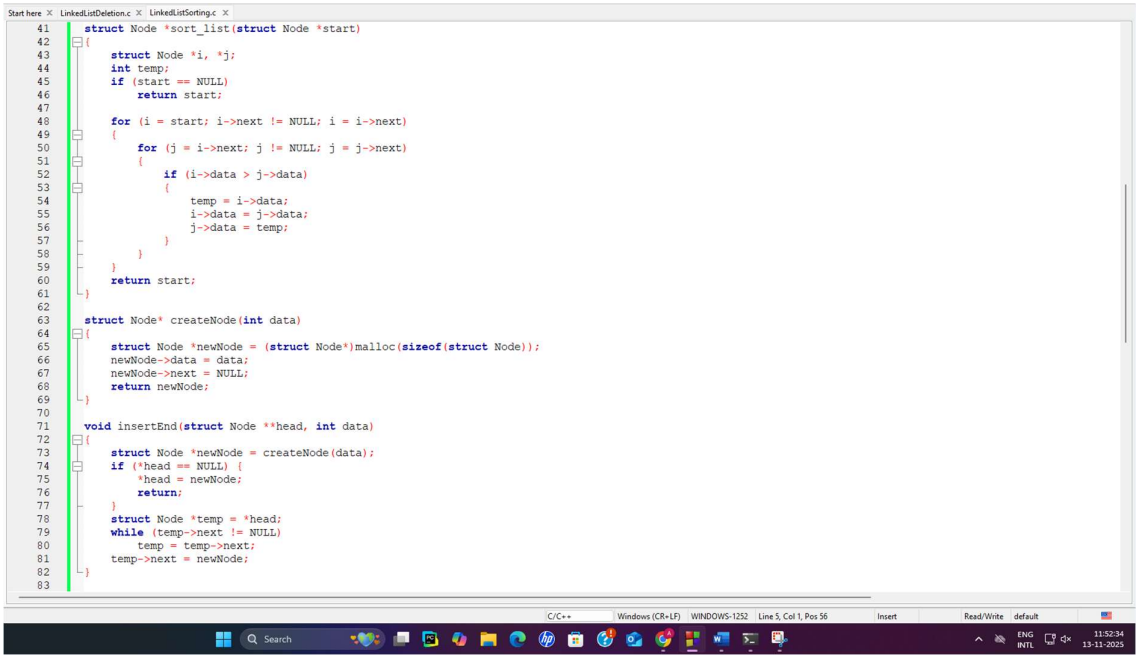

D:\BF24CS067\LinkedListDel
--- Linked List Deletion Menu ---
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 1
Enter data: 11
--- Linked List Deletion Menu ---
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 1
Enter data: 12
--- Linked List Deletion Menu ---
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 1
Enter data: 23
--- Linked List Deletion Menu ---
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 5
Linked List: 11 -> 12 -> 23 -> NULL
--- Linked List Deletion Menu ---
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 2
Deleted element: 11
Finance headline
India reported 0...
Search
ENG
INTL
11:21:29
13-11-2025


D:\BF24CS067\LinkedListDel
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 5
Linked List: 11 -> 12 -> 23 -> NULL
--- Linked List Deletion Menu ---
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 2
Deleted element: 11
--- Linked List Deletion Menu ---
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 3
Deleted element: 23
--- Linked List Deletion Menu ---
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 5
Linked List: 12 -> NULL
--- Linked List Deletion Menu ---
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 6
Process returned 0 (0x0) execution time : 135.353 s
Press any key to continue.
Finance headline
India reported 0...
Search
ENG
INTL
11:22:04
13-11-2025

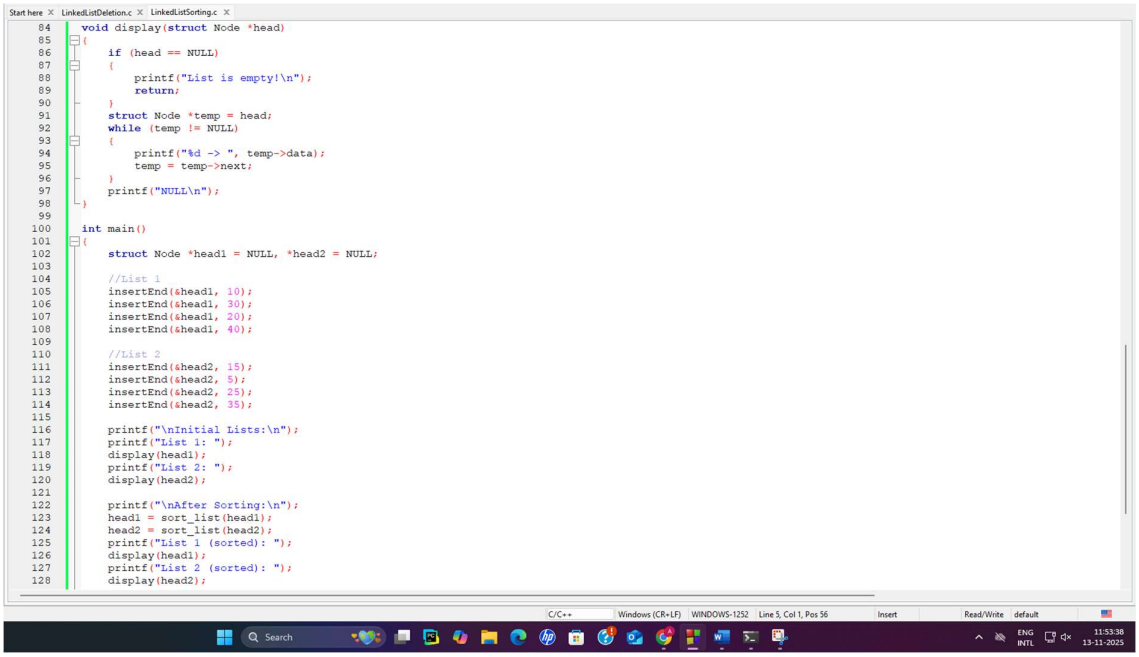
b). SLL - (sort, reverse, concatenation)



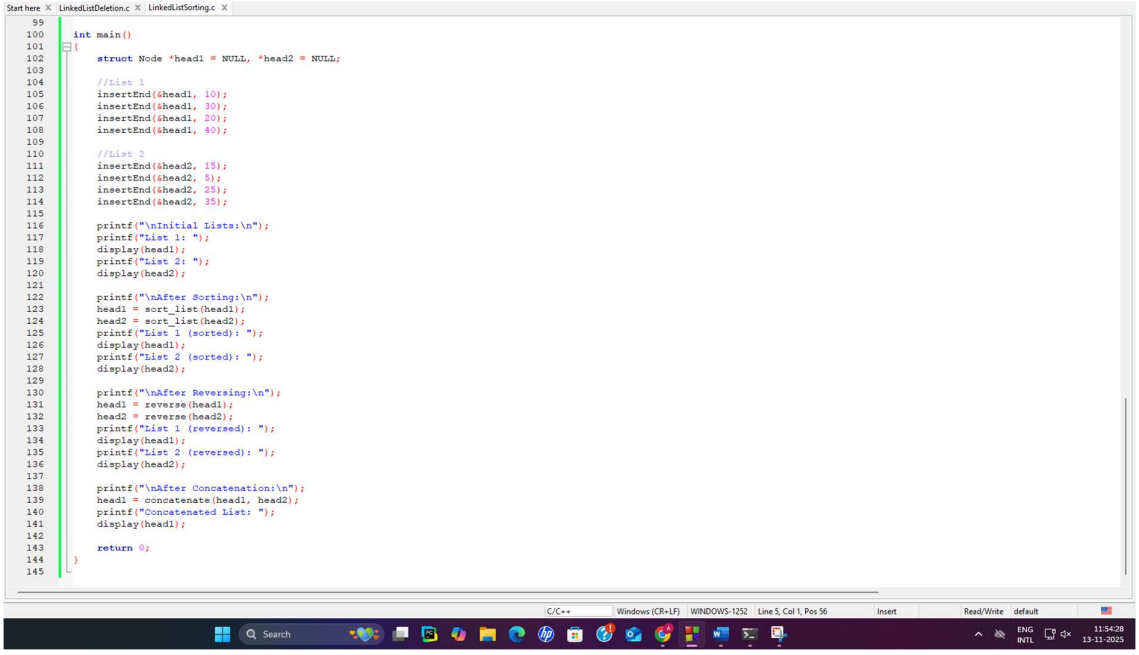
```c
#include <stdio.h>
#include <stdlib.h>

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

struct Node *reverse(struct Node *start)
{
    struct Node *prev = NULL;
    struct Node *curr = start;
    struct Node *next;
    while (curr != NULL)
    {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}

struct Node *concatenate(struct Node *start1, struct Node *start2)
{
    struct Node *ptr;
    if (start1 == NULL)
    {
        return start2;
    }
    ptr = start1;
    while (ptr->next != NULL)
    {
        ptr = ptr->next;
    }
    ptr->next = start2;
    return start1;
}

struct Node *sort_list(struct Node *start)
{
    struct Node *i, *j;
```



```c
struct Node *sort_list(struct Node *start)
{
    struct Node *i, *j;
    int temp;
    if (start == NULL)
        return start;

    for (i = start; i->next != 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;
            }
        }
    }
    return start;
}

struct Node* createNode(int data)
{
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void insertEnd(struct Node **head, int data)
{
    struct Node *newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node *temp = *head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
}
```

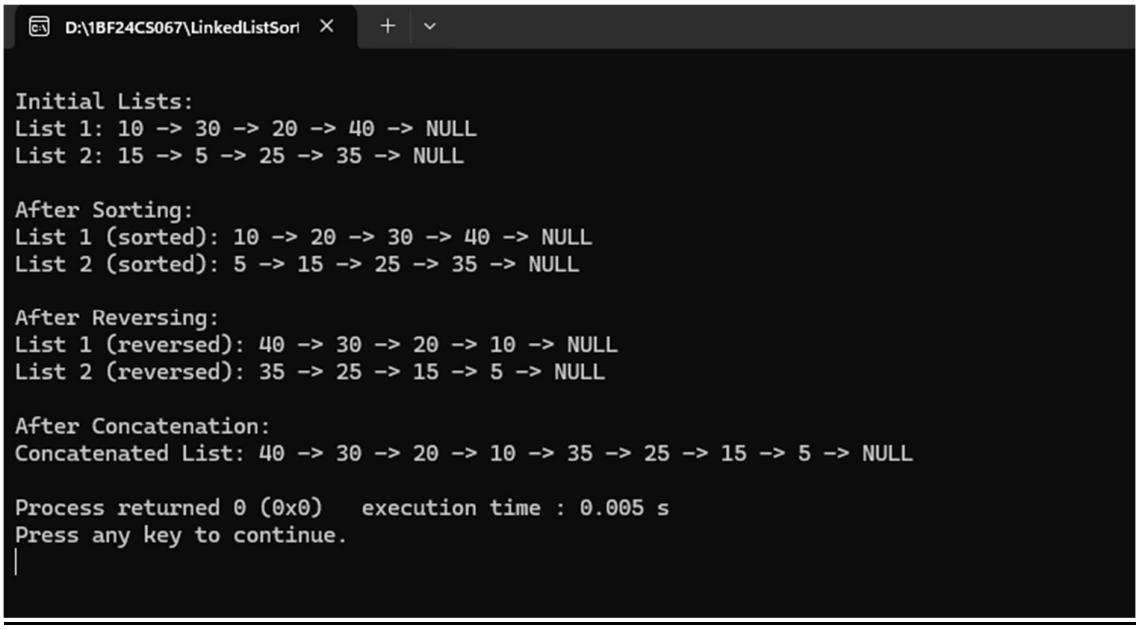


```c
void display(struct Node *head)
{
    if (head == NULL)
    {
        printf("List is empty!\n");
        return;
    }
    struct Node *temp = head;
    while (temp != NULL)
    {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main()
{
    struct Node *head1 = NULL, *head2 = NULL;

    //List 1
    insertEnd(&head1, 10);
    insertEnd(&head1, 30);
    insertEnd(&head1, 20);
    insertEnd(&head1, 40);

    //List 2
    insertEnd(&head2, 15);
    insertEnd(&head2, 5);
    insertEnd(&head2, 25);
    insertEnd(&head2, 35);

    printf("\nInitial Lists:\n");
    printf("List 1: ");
    display(head1);
    printf("List 2: ");
    display(head2);

    printf("\nAfter Sorting:\n");
    head1 = sort_list(head1);
    head2 = sort_list(head2);
    printf("List 1 (sorted): ");
    display(head1);
    printf("List 2 (sorted): ");
    display(head2);
```



```c
int main()
{
    struct Node *head1 = NULL, *head2 = NULL;

    //List 1
    insertEnd(&head1, 10);
    insertEnd(&head1, 30);
    insertEnd(&head1, 20);
    insertEnd(&head1, 40);

    //List 2
    insertEnd(&head2, 15);
    insertEnd(&head2, 5);
    insertEnd(&head2, 25);
    insertEnd(&head2, 35);

    printf("\nInitial Lists:\n");
    printf("List 1: ");
    display(head1);
    printf("List 2: ");
    display(head2);

    printf("\nAfter Sorting:\n");
    head1 = sort_list(head1);
    head2 = sort_list(head2);
    printf("List 1 (sorted): ");
    display(head1);
    printf("List 2 (sorted): ");
    display(head2);

    printf("\nAfter Reversing:\n");
    head1 = reverse(head1);
    head2 = reverse(head2);
    printf("List 1 (reversed): ");
    display(head1);
    printf("List 2 (reversed): ");
    display(head2);

    printf("\nAfter Concatenation:\n");
    head1 = concatenate(head1, head2);
    printf("Concatenated List: ");
    display(head1);

    return 0;
}
```

OUTPUT:


D:\1BF24CS067\LinkedListSor
Initial Lists:
List 1: 10 -> 30 -> 20 -> 40 -> NULL
List 2: 15 -> 5 -> 25 -> 35 -> NULL
After Sorting:
List 1 (sorted): 10 -> 20 -> 30 -> 40 -> NULL
List 2 (sorted): 5 -> 15 -> 25 -> 35 -> NULL
After Reversing:
List 1 (reversed): 40 -> 30 -> 20 -> 10 -> NULL
List 2 (reversed): 35 -> 25 -> 15 -> 5 -> NULL
After Concatenation:
Concatenated List: 40 -> 30 -> 20 -> 10 -> 35 -> 25 -> 15 -> 5 -> NULL
Process returned 0 (0x0) execution time : 0.005 s
Press any key to continue.