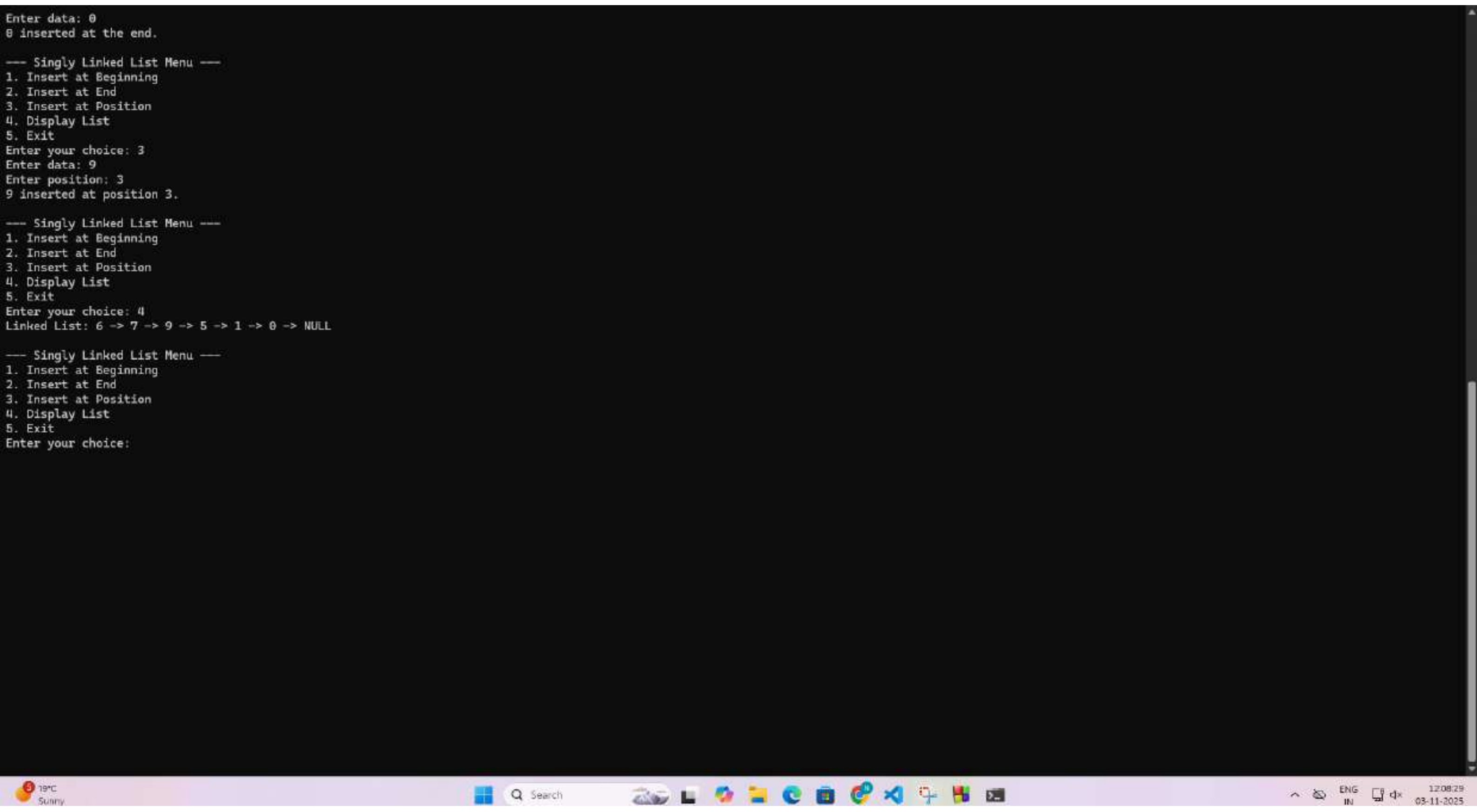
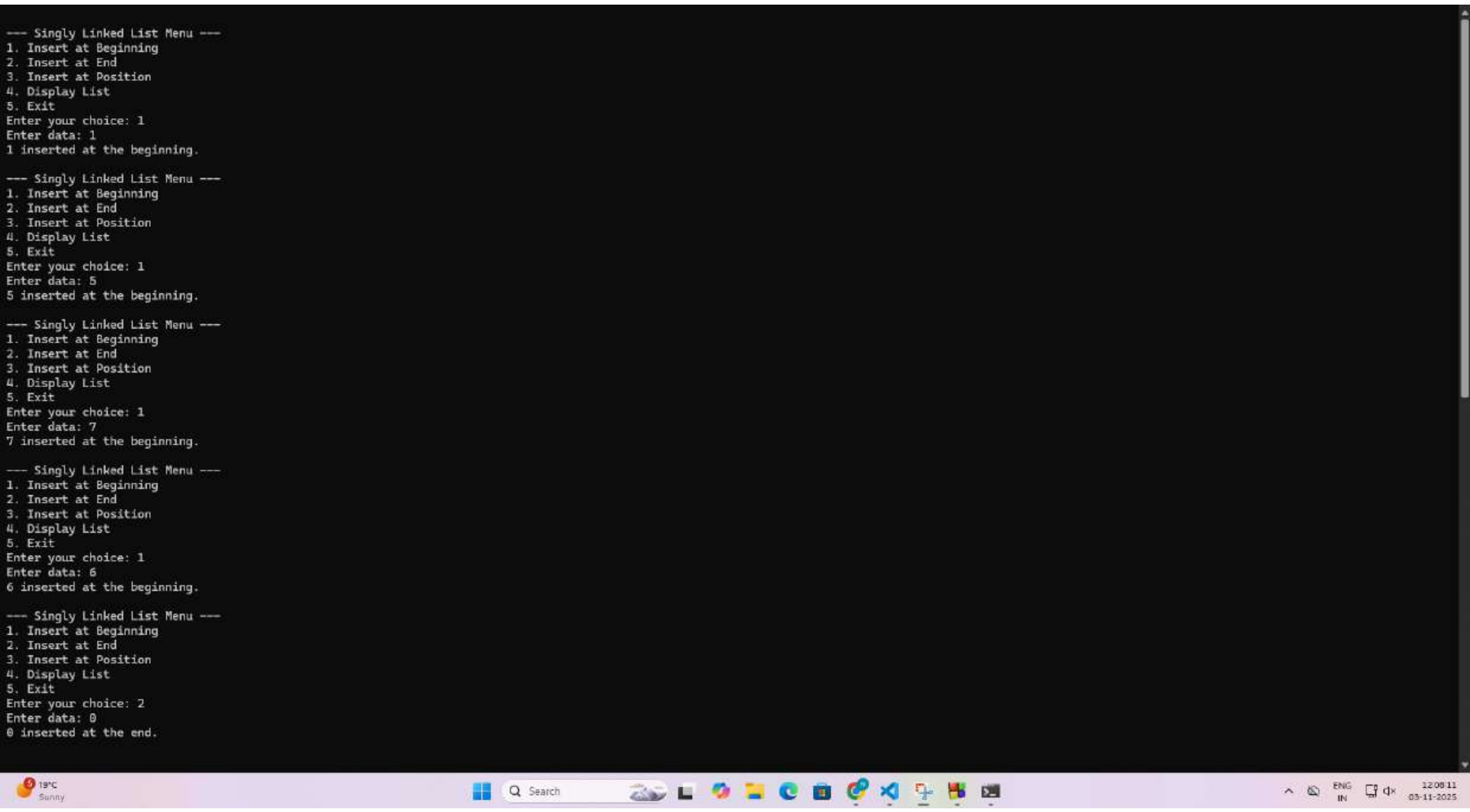
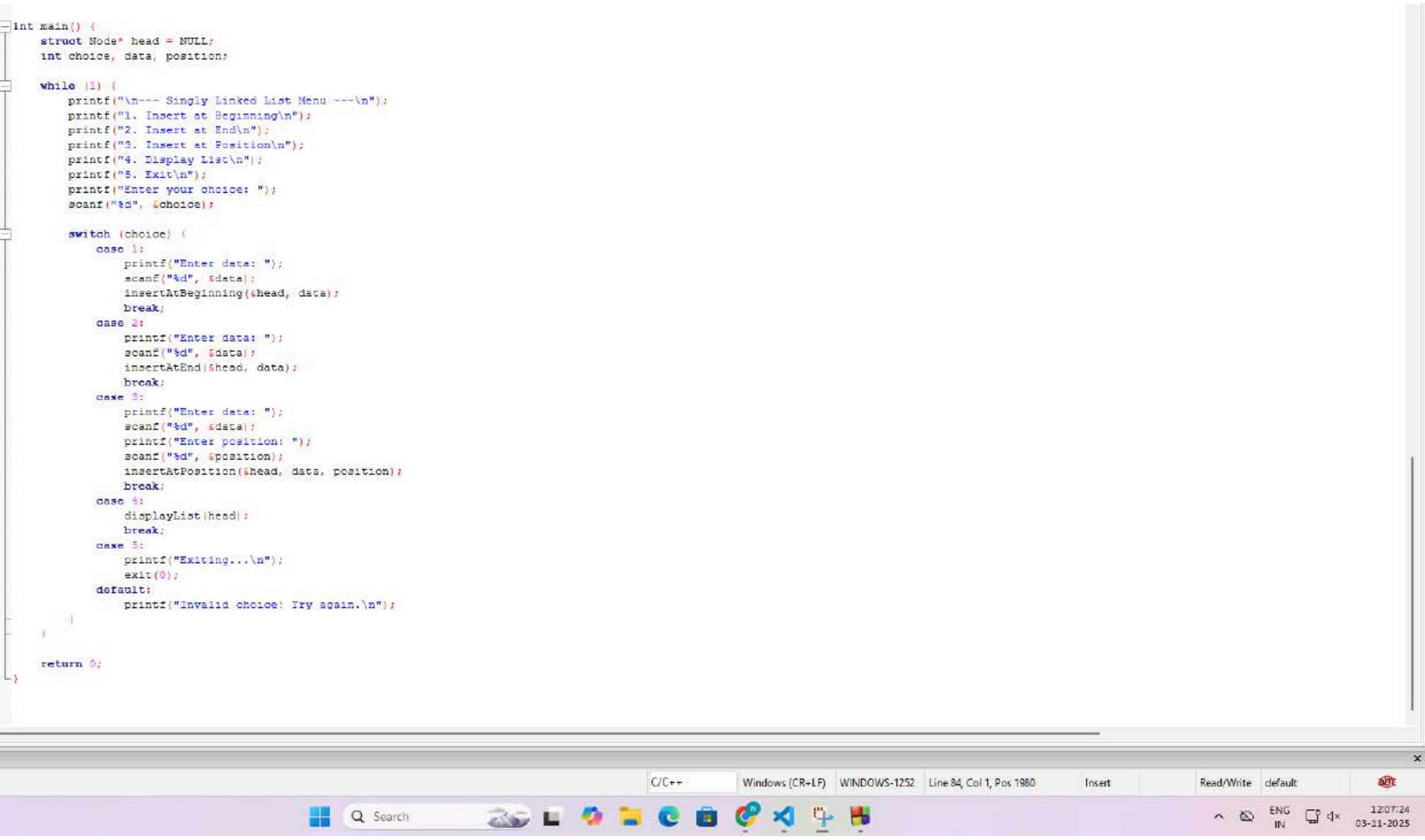


```
Enter data: 0
0 inserted at the end.

--- Singly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display List
5. Exit
Enter your choice: 3
Enter data: 9
Enter position: 3
9 inserted at position 3.

--- Singly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display List
5. Exit
Enter your choice: 4
Linked List: 6 -> 7 -> 9 -> 5 -> 1 -> 0 -> NULL

--- Singly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display List
5. Exit
Enter your choice:
```

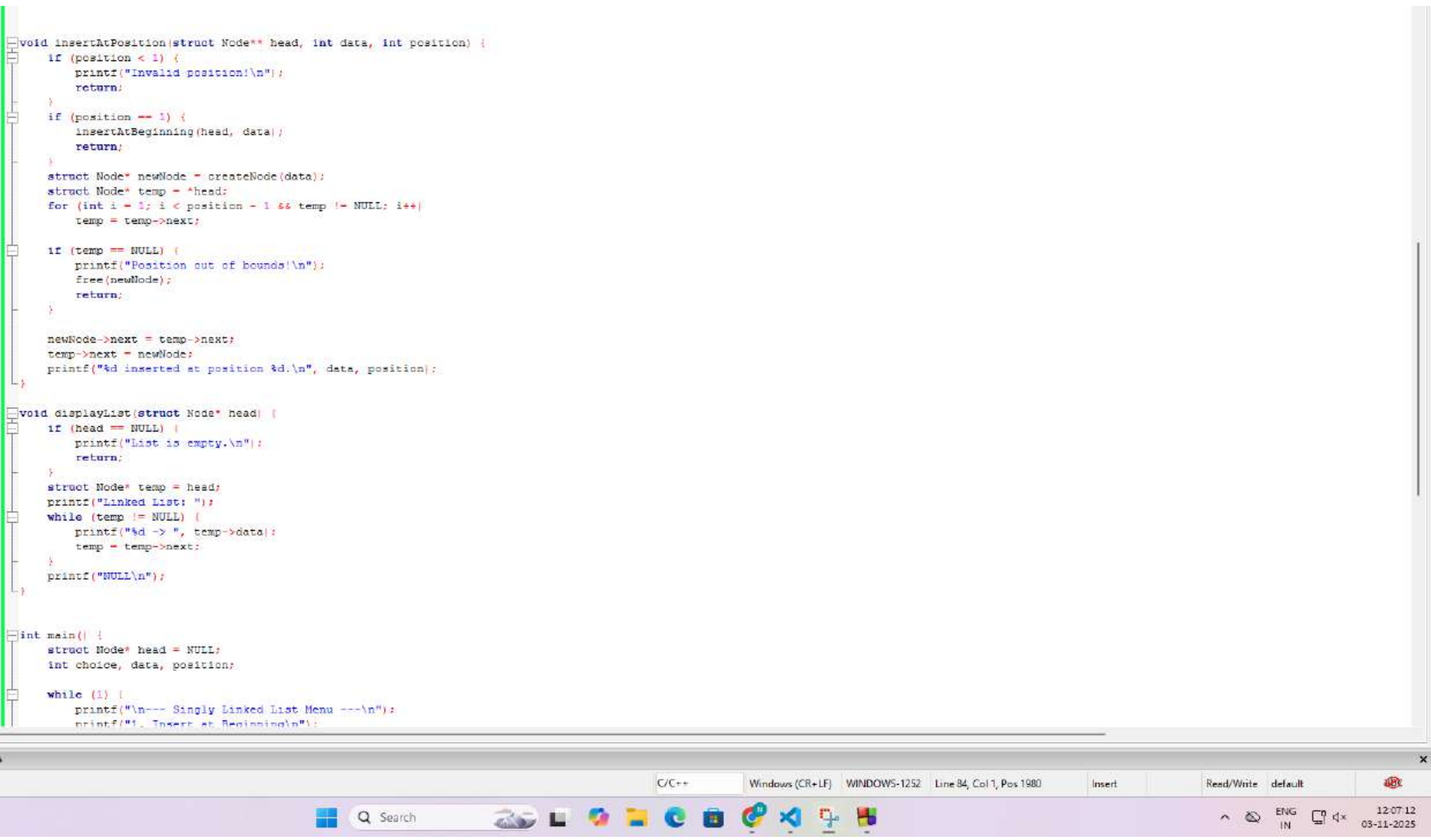


```
--- Singly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display List
5. Exit
Enter your choice: 1
Enter data: 1
1 inserted at the beginning.

--- Singly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display List
5. Exit
Enter your choice: 1
Enter data: 5
5 inserted at the beginning.

--- Singly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display List
5. Exit
Enter your choice: 1
Enter data: 7
7 inserted at the beginning.

--- Singly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display List
5. Exit
Enter your choice: 1
Enter data: 6
6 inserted at the beginning.

--- Singly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Display List
5. Exit
Enter your choice: 2
Enter data: 0
0 inserted at the end.
```

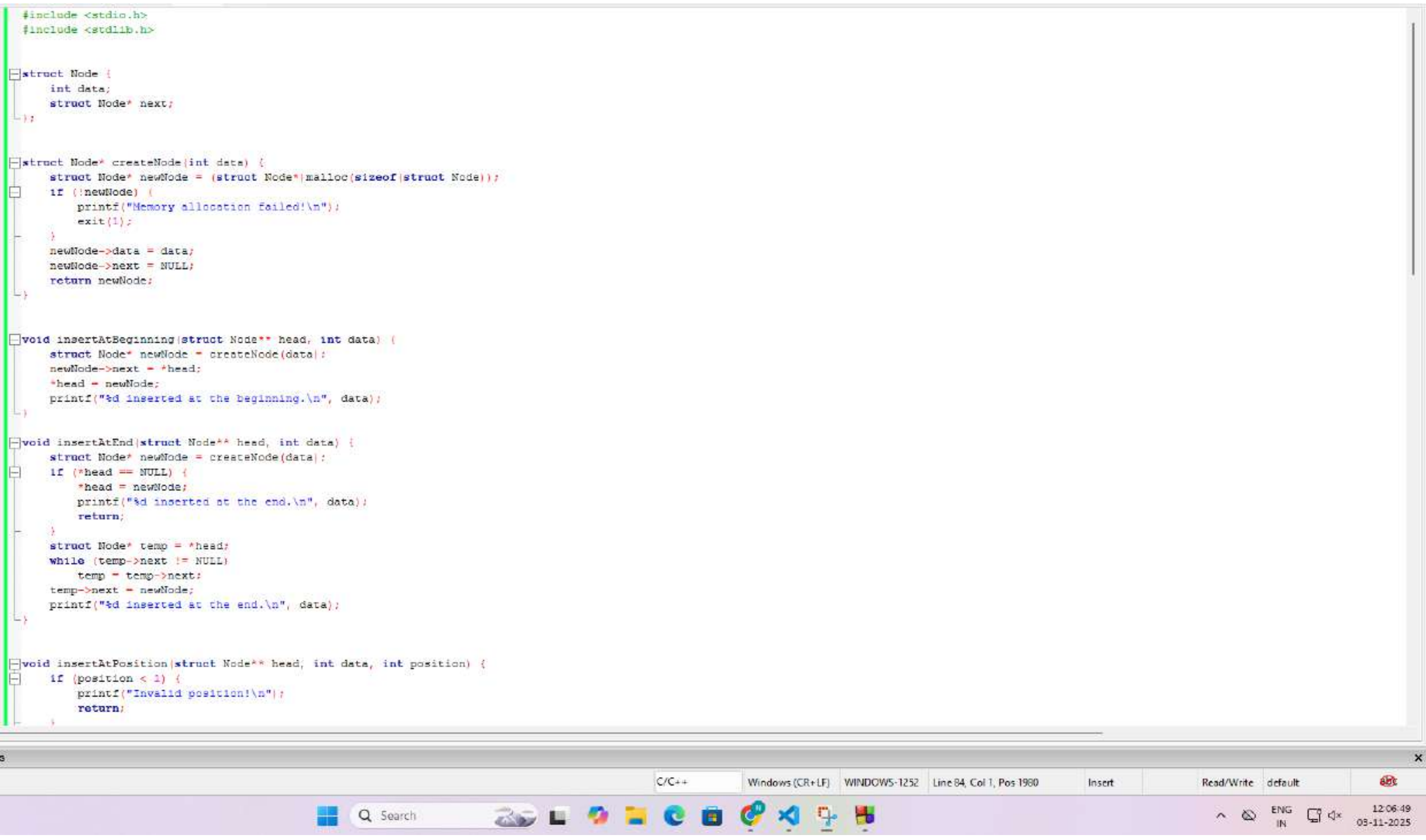


```c
int main() {
    struct Node* head = NULL;
    int choice, data, position;

    while (1) {
        printf("\n--- Singly Linked List Menu ---\n");
        printf("1. Insert at Beginning\n");
        printf("2. Insert at End\n");
        printf("3. Insert at Position\n");
        printf("4. Display List\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter data: ");
                scanf("%d", &data);
                insertAtBeginning(&head, data);
                break;
            case 2:
                printf("Enter data: ");
                scanf("%d", &data);
                insertAtEnd(&head, data);
                break;
            case 3:
                printf("Enter data: ");
                scanf("%d", &data);
                printf("Enter position: ");
                scanf("%d", &position);
                insertAtPosition(&head, data, position);
                break;
            case 4:
                displayList(head);
                break;
            case 5:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice! Try again.\n");
        }
    }

    return 0;
}
```

C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 84, Col 1, Pos 1980 | Insert | Read/Write | default



```c
void insertAtPosition(struct Node** head, int data, int position) {
    if (position < 1) {
        printf("Invalid position!\n");
        return;
    }
    if (position == 1) {
        insertAtBeginning(head, data);
        return;
    }
    struct Node* newNode = createNode(data);
    struct Node* temp = *head;
    for (int i = 1; i < position - 1 && temp != NULL; i++)
        temp = temp->next;

    if (temp == NULL) {
        printf("Position out of bounds!\n");
        free(newNode);
        return;
    }

    newNode->next = temp->next;
    temp->next = newNode;
    printf("%d inserted at position %d.\n", data, position);
}

void displayList(struct Node* head) {
    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() {
    struct Node* head = NULL;
    int choice, data, position;

    while (1) {
        printf("\n--- Singly Linked List Menu ---\n");
        printf("1. Insert at Beginning\n");
```



```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));
    if (!newNode) {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}


void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *head;
    *head = newNode;
    printf("%d inserted at the beginning.\n", data);
}

void insertAtEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        printf("%d inserted at the end.\n", data);
        return;
    }
    struct Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
    printf("%d inserted at the end.\n", data);
}


void insertAtPosition(struct Node** head, int data, int position) {
    if (position < 1) {
        printf("Invalid position!\n");
        return;
    }
```

C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 84, Col 1, Pos 1980 | Insert | Read/Write | default

ENG IN 12:06:49 03-11-2025