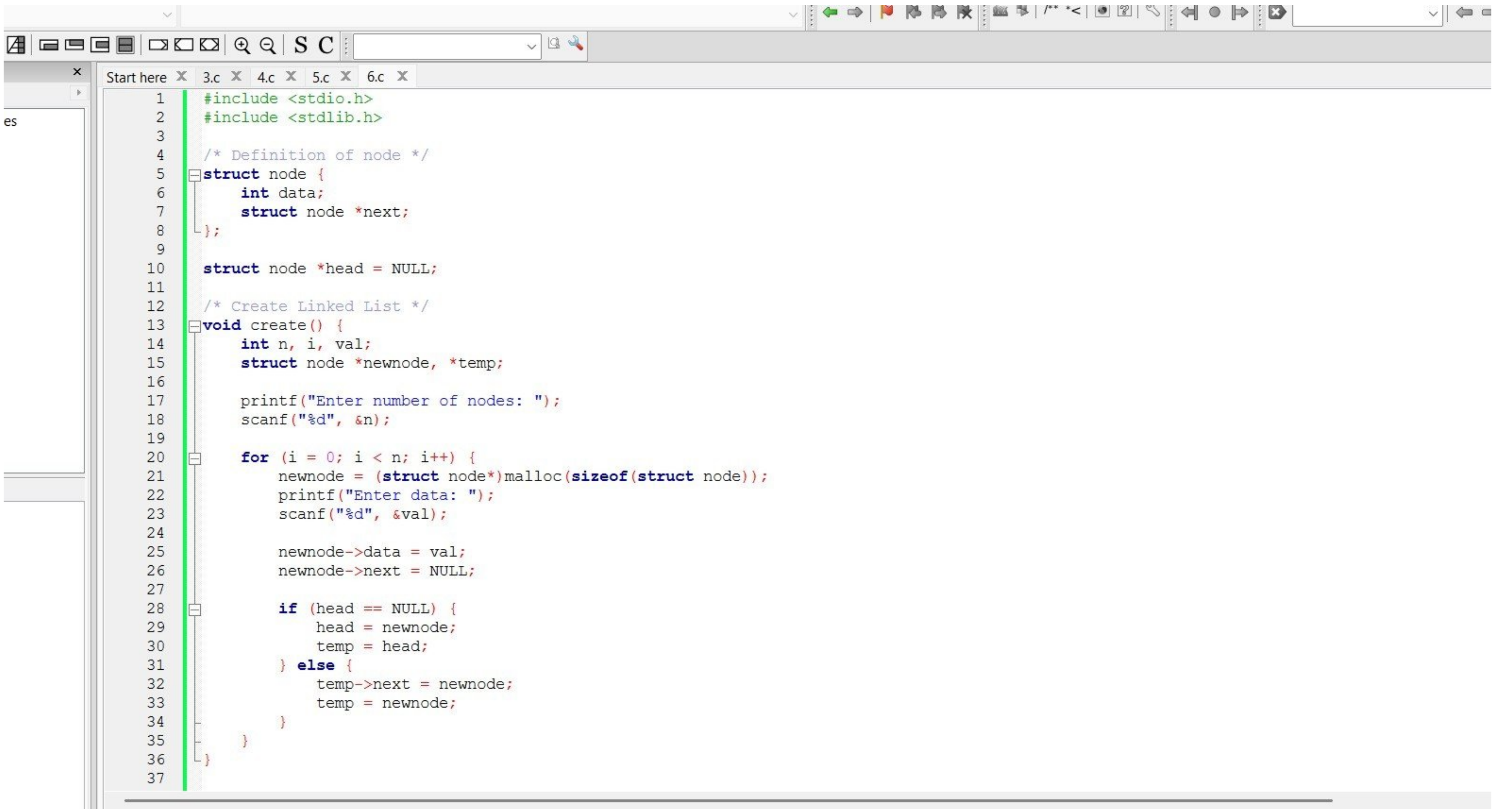
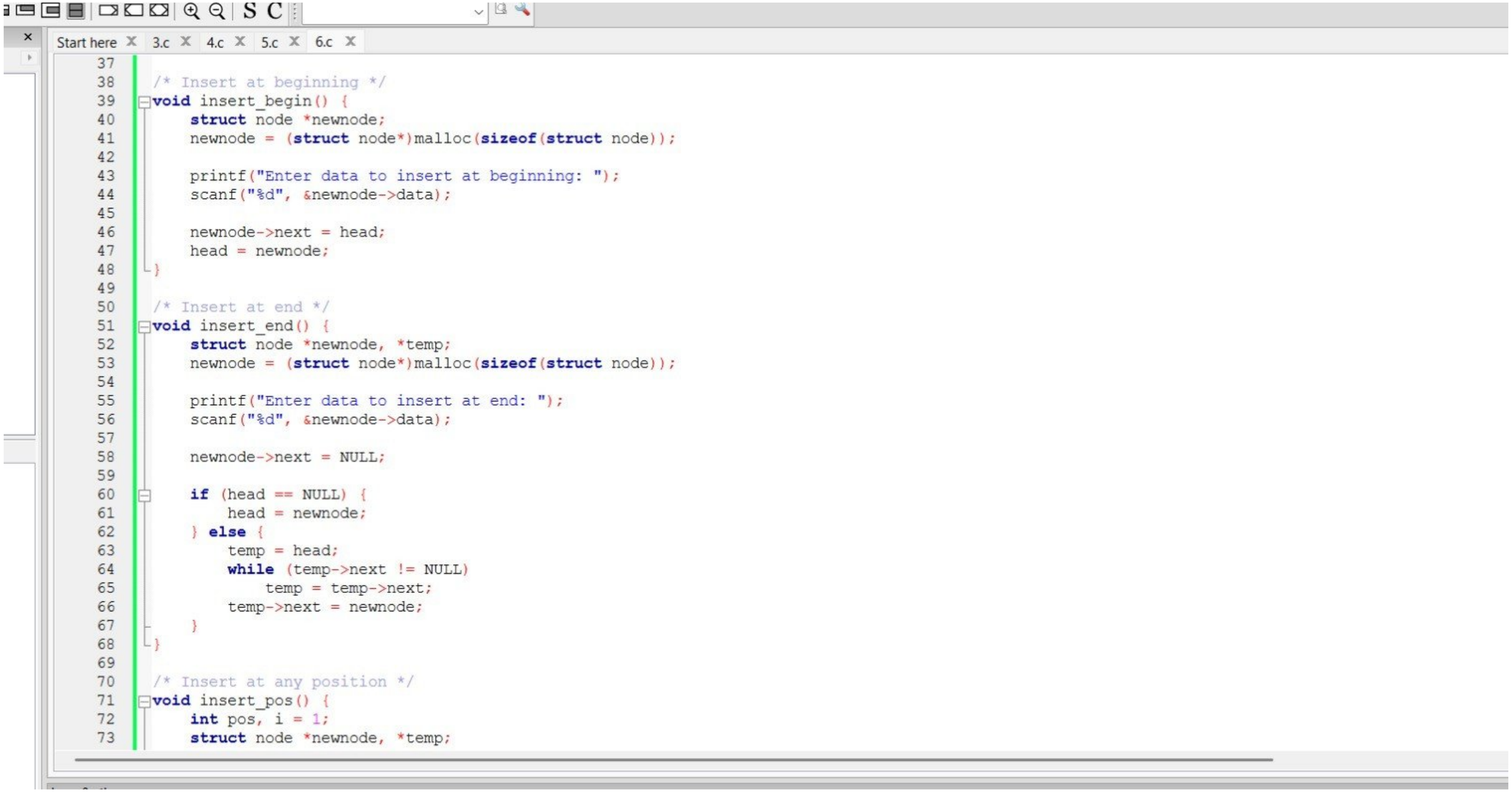
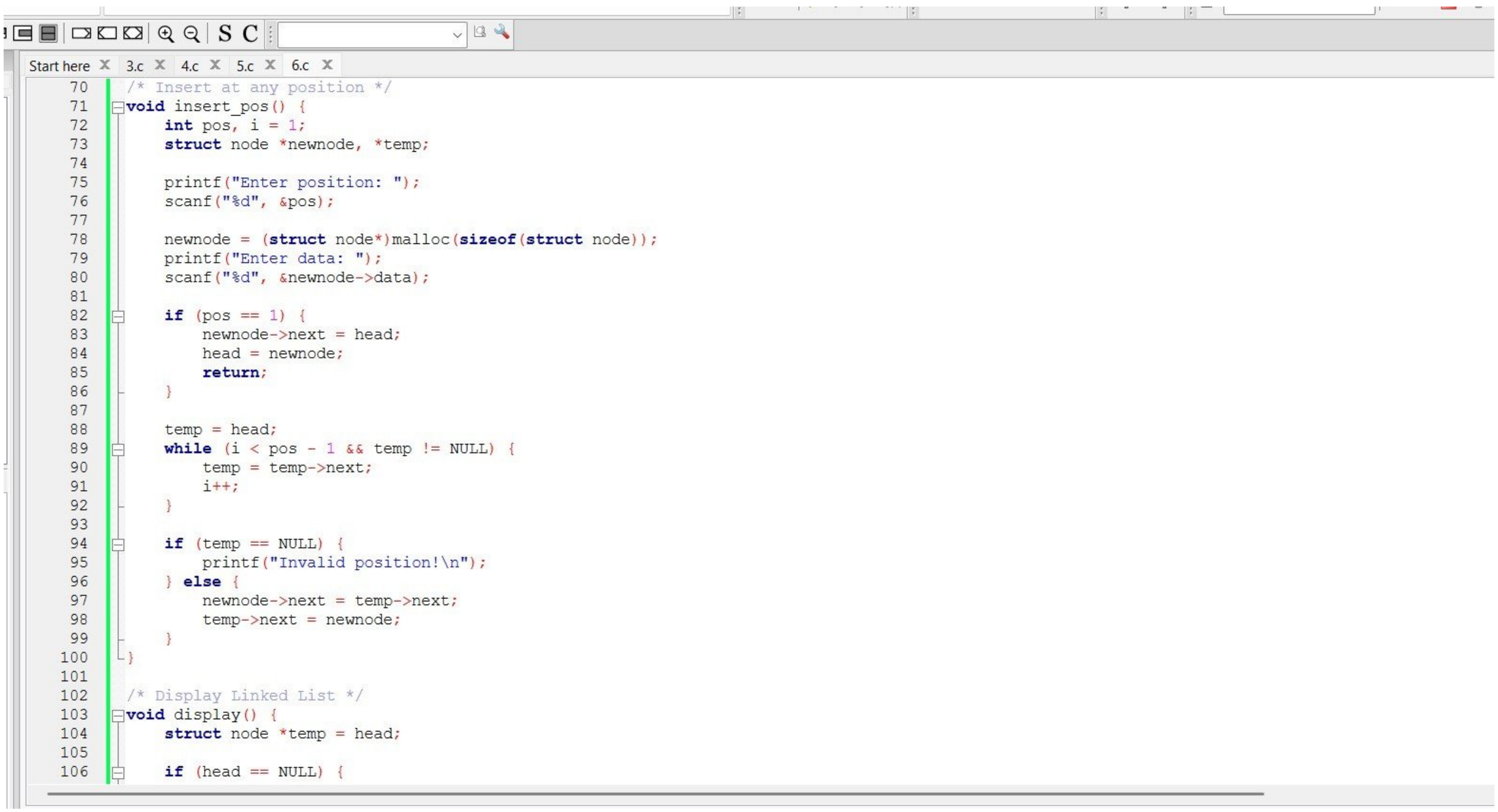
3	2	5	a) WAP to simulate the working of a queue of integers using an array. Provide the following operations: Insert, Delete, Display The program should print appropriate messages for queue empty and queue overflow conditions
		5	b) WAP to simulate the working of a circular queue of integers using an array. Provide the following operations: Insert, Delete & Display The program should print appropriate messages for queue empty and queue overflow conditions



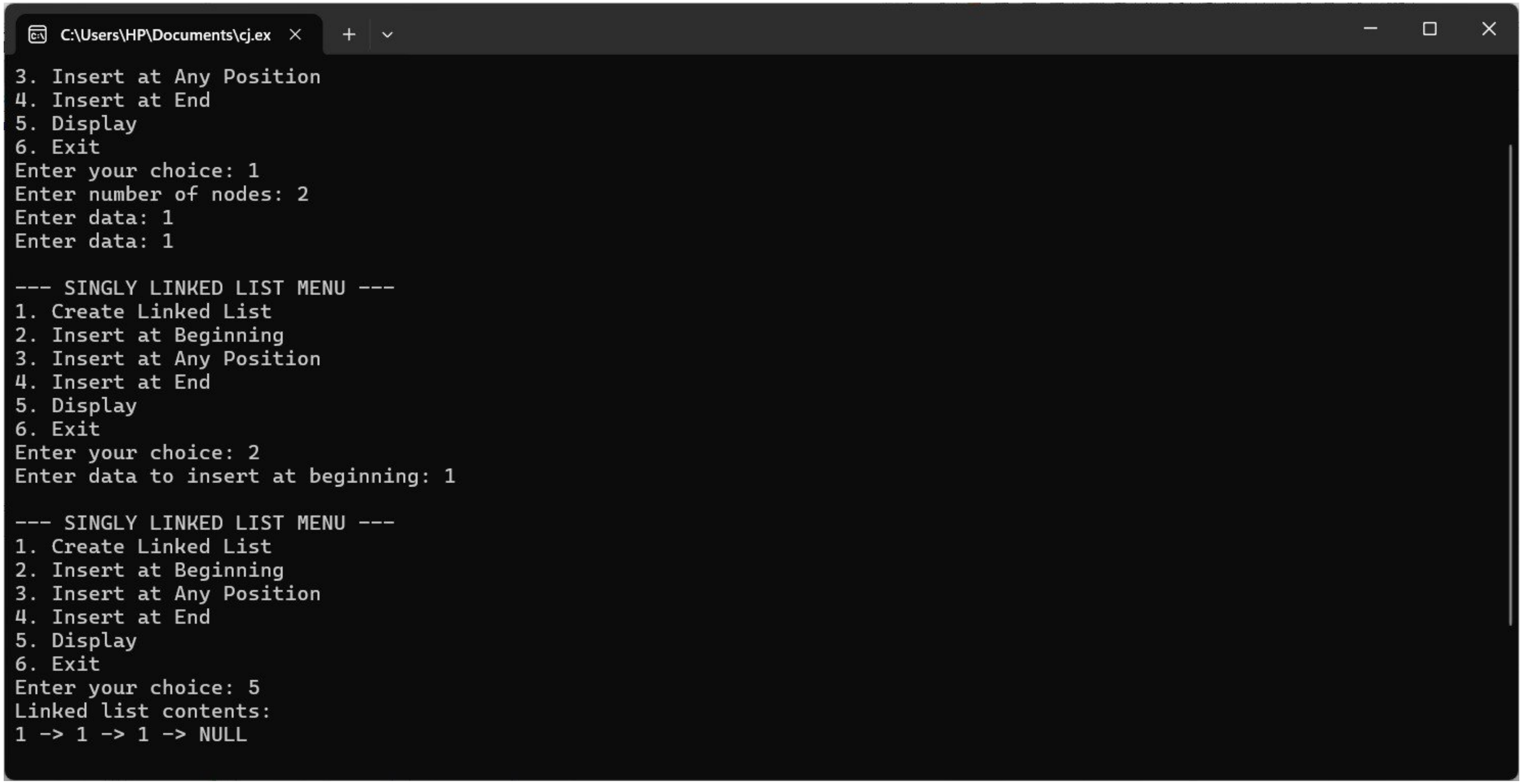
```c
#include <stdio.h>
#include <stdlib.h>

/* Definition of node */
struct node {
    int data;
    struct node *next;
};

struct node *head = NULL;

/* Create Linked List */
void create() {
    int n, i, val;
    struct node *newnode, *temp;

    printf("Enter number of nodes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        newnode = (struct node*)malloc(sizeof(struct node));
        printf("Enter data: ");
        scanf("%d", &val);

        newnode->data = val;
        newnode->next = NULL;

        if (head == NULL) {
            head = newnode;
            temp = head;
        } else {
            temp->next = newnode;
            temp = newnode;
        }
    }
}

```



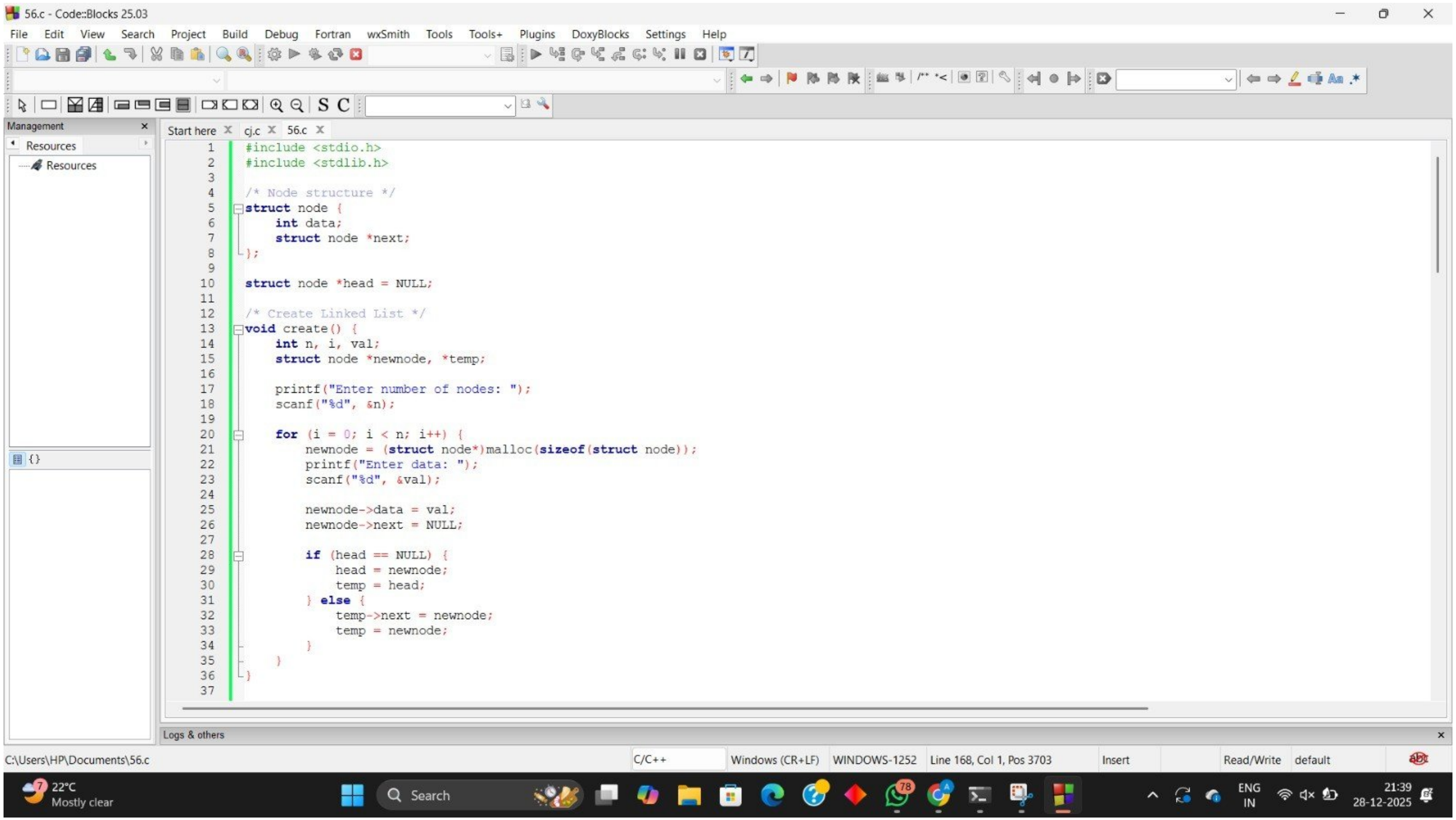
```c

/* Insert at beginning */
void insert_begin() {
    struct node *newnode;
    newnode = (struct node*)malloc(sizeof(struct node));

    printf("Enter data to insert at beginning: ");
    scanf("%d", &newnode->data);

    newnode->next = head;
    head = newnode;
}

/* Insert at end */
void insert_end() {
    struct node *newnode, *temp;
    newnode = (struct node*)malloc(sizeof(struct node));

    printf("Enter data to insert at end: ");
    scanf("%d", &newnode->data);

    newnode->next = NULL;

    if (head == NULL) {
        head = newnode;
    } else {
        temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newnode;
    }
}

/* Insert at any position */
void insert_pos() {
    int pos, i = 1;
    struct node *newnode, *temp;
```



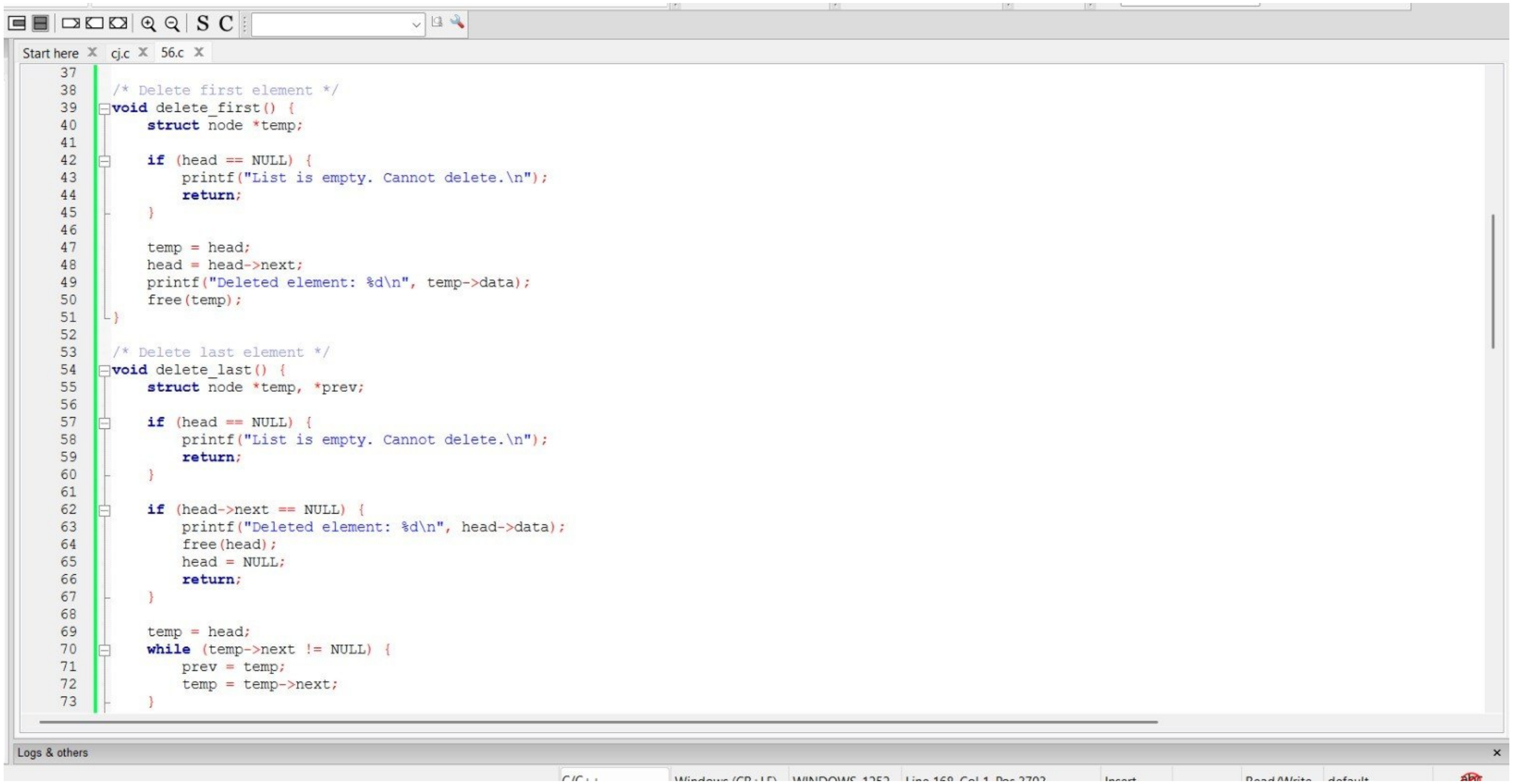
```c
/* Insert at any position */
void insert_pos() {
    int pos, i = 1;
    struct node *newnode, *temp;

    printf("Enter position: ");
    scanf("%d", &pos);

    newnode = (struct node*)malloc(sizeof(struct node));
    printf("Enter data: ");
    scanf("%d", &newnode->data);

    if (pos == 1) {
        newnode->next = head;
        head = newnode;
        return;
    }

    temp = head;
    while (i < pos - 1 && temp != NULL) {
        temp = temp->next;
        i++;
    }

    if (temp == NULL) {
        printf("Invalid position!\n");
    } else {
        newnode->next = temp->next;
        temp->next = newnode;
    }
}

/* Display Linked List */
void display() {
    struct node *temp = head;

    if (head == NULL) {
```



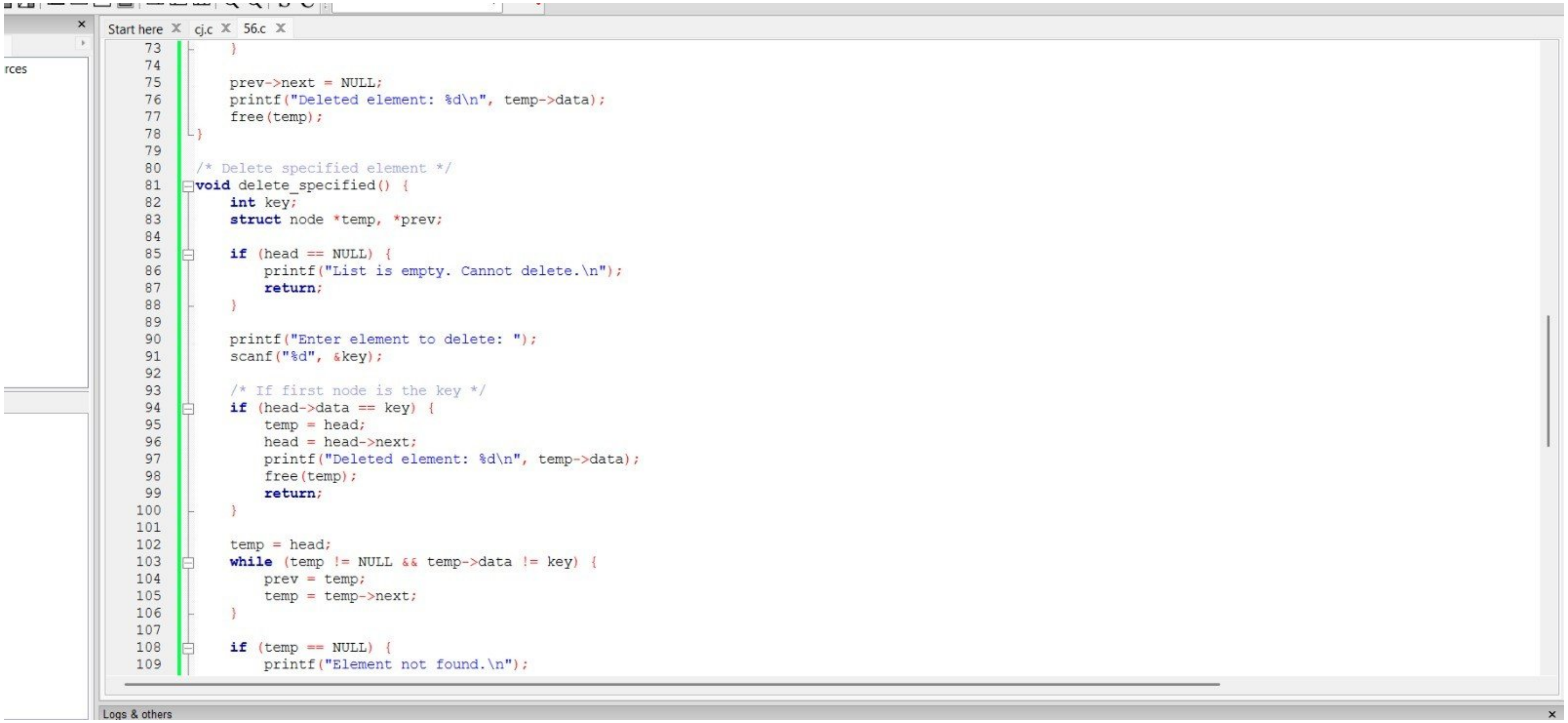
```
3. Insert at Any Position
4. Insert at End
5. Display
6. Exit
Enter your choice: 1
Enter number of nodes: 2
Enter data: 1
Enter data: 1

--- SINGLY LINKED LIST MENU ---
1. Create Linked List
2. Insert at Beginning
3. Insert at Any Position
4. Insert at End
5. Display
6. Exit
Enter your choice: 2
Enter data to insert at beginning: 1

--- SINGLY LINKED LIST MENU ---
1. Create Linked List
2. Insert at Beginning
3. Insert at Any Position
4. Insert at End
5. Display
6. Exit
Enter your choice: 5
Linked list contents:
1 -> 1 -> 1 -> NULL
```


56.c - Code::Blocks 25.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Resources
Start here
cj.c
56.c
Logs & others
C:\Users\HP\Documents\56.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 168, Col 1, Pos 3703
Insert
Read/Write
default
22°C
Mostly clear
Search
ENG
IN
21:39
28-12-2025

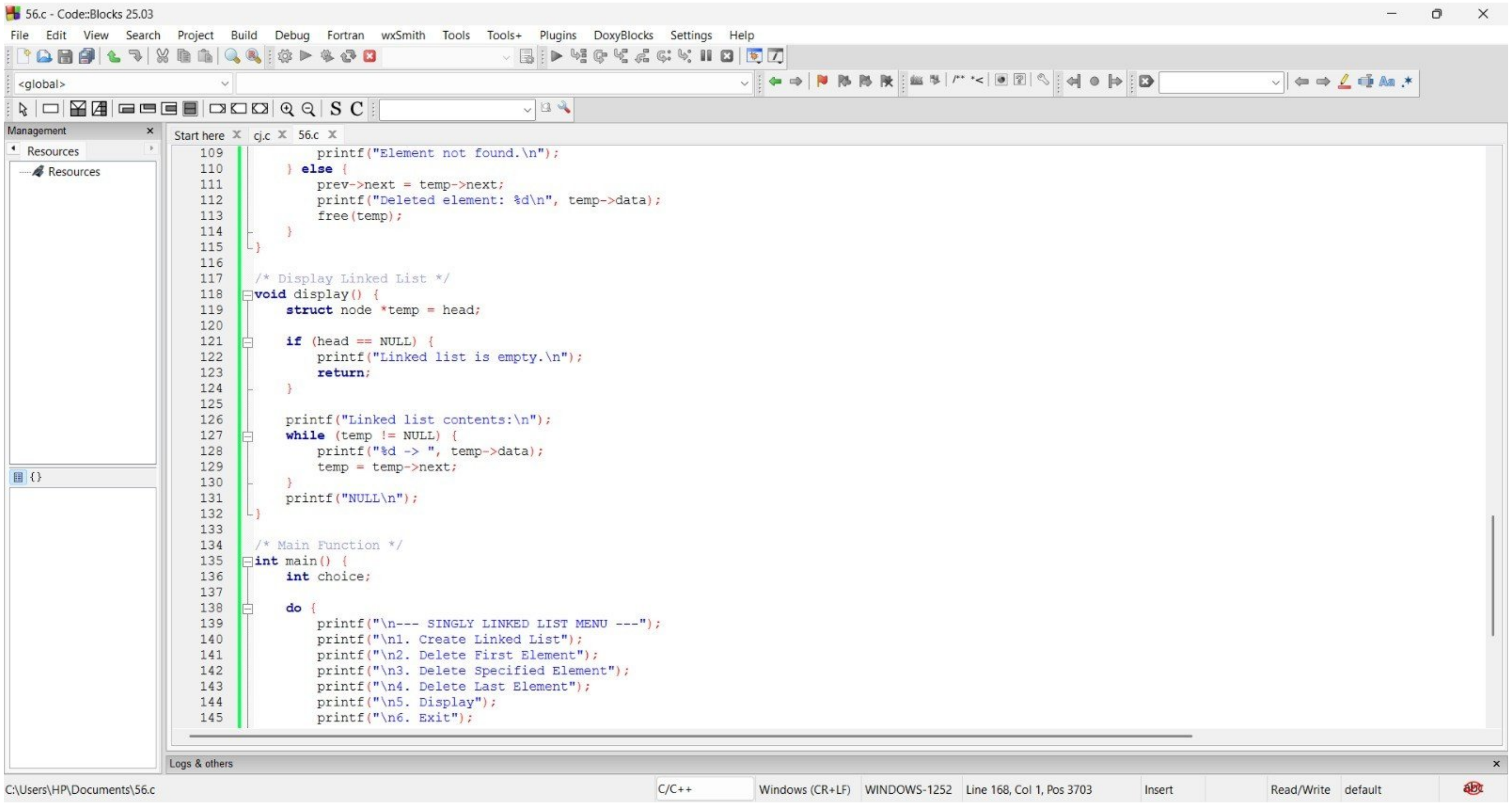
```c
#include <stdio.h>
#include <stdlib.h>

/* Node structure */
struct node {
    int data;
    struct node *next;
};

struct node *head = NULL;

/* Create Linked List */
void create() {
    int n, i, val;
    struct node *newnode, *temp;

    printf("Enter number of nodes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        newnode = (struct node*)malloc(sizeof(struct node));
        printf("Enter data: ");
        scanf("%d", &val);

        newnode->data = val;
        newnode->next = NULL;

        if (head == NULL) {
            head = newnode;
            temp = head;
        } else {
            temp->next = newnode;
            temp = newnode;
        }
    }
}

```



```c

/* Delete first element */
void delete_first() {
    struct node *temp;

    if (head == NULL) {
        printf("List is empty. Cannot delete.\n");
        return;
    }

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

/* Delete last element */
void delete_last() {
    struct node *temp, *prev;

    if (head == NULL) {
        printf("List is empty. Cannot delete.\n");
        return;
    }

    if (head->next == NULL) {
        printf("Deleted element: %d\n", head->data);
        free(head);
        head = NULL;
        return;
    }

    temp = head;
    while (temp->next != NULL) {
        prev = temp;
        temp = temp->next;
    }
```

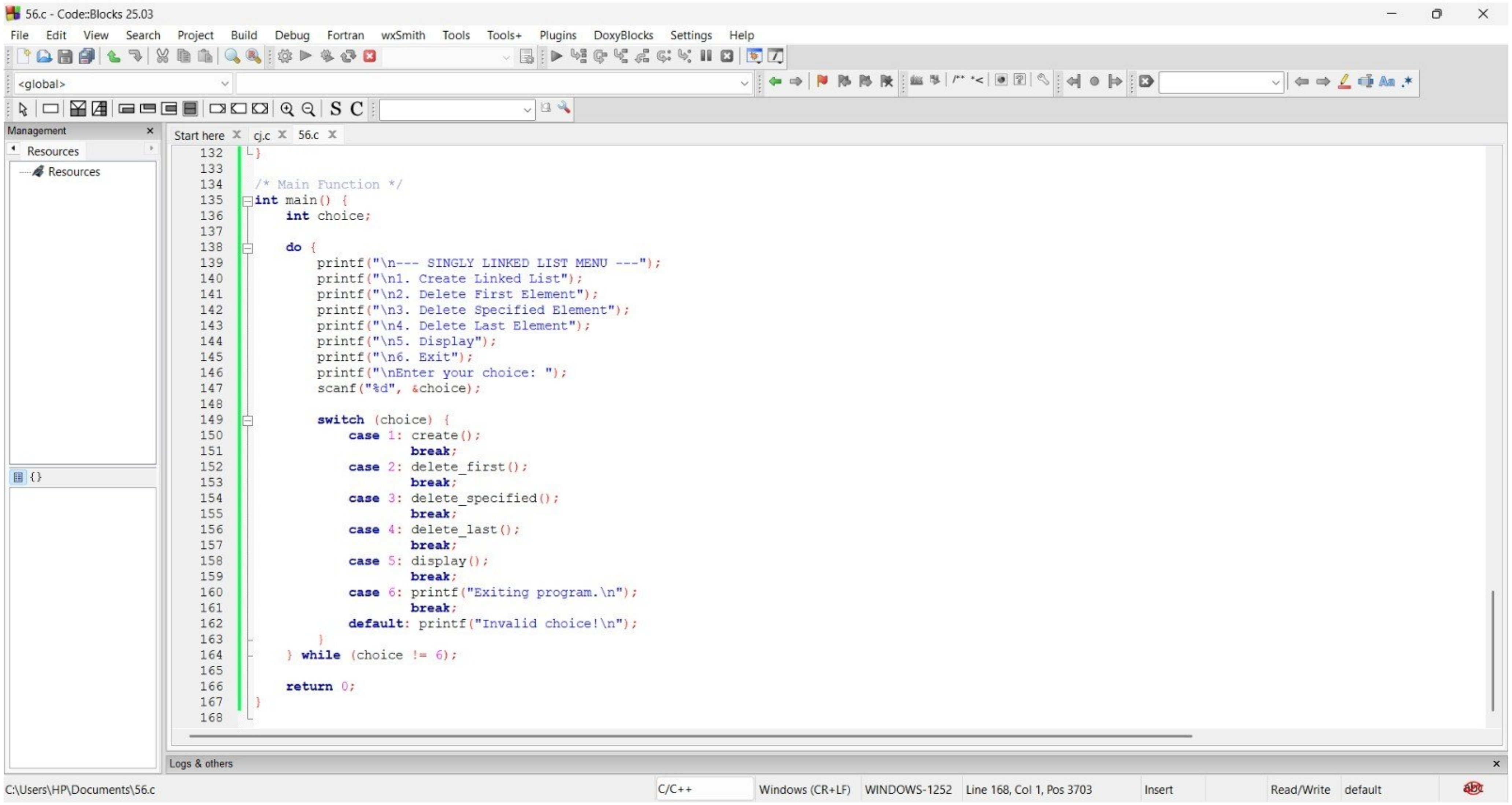


```c
    }

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

/* Delete specified element */
void delete_specified() {
    int key;
    struct node *temp, *prev;

    if (head == NULL) {
        printf("List is empty. Cannot delete.\n");
        return;
    }

    printf("Enter element to delete: ");
    scanf("%d", &key);

    /* If first node is the key */
    if (head->data == key) {
        temp = head;
        head = head->next;
        printf("Deleted element: %d\n", temp->data);
        free(temp);
        return;
    }

    temp = head;
    while (temp != NULL && temp->data != key) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Element not found.\n");
```

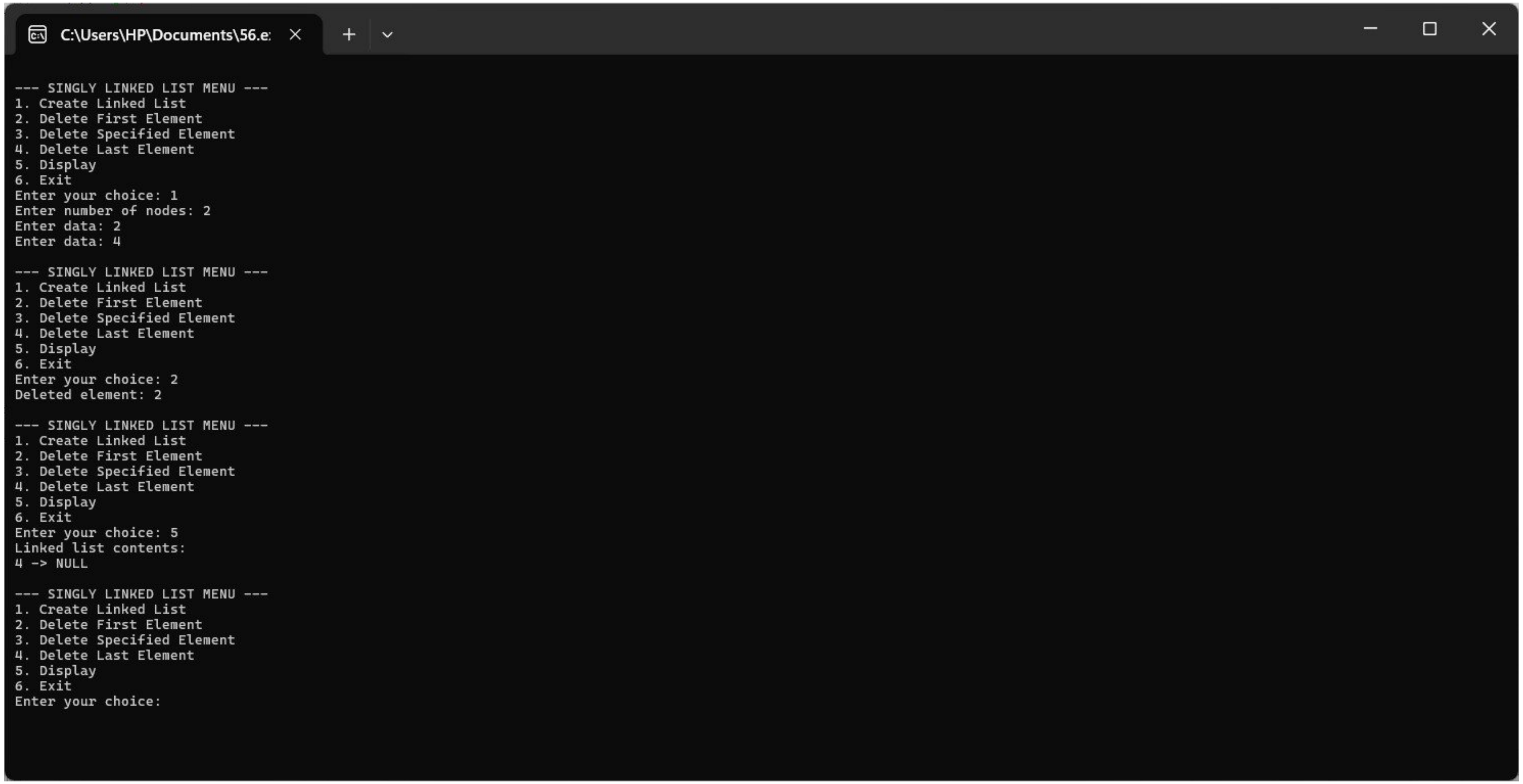

56.c - Code::Blocks 25.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Resources
Start here
cj.c
56.c
Logs & others

```
            printf("Element not found.\n");
        } else {
            prev->next = temp->next;
            printf("Deleted element: %d\n", temp->data);
            free(temp);
        }
}

/* Display Linked List */
void display() {
    struct node *temp = head;

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

    printf("Linked list contents:\n");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

/* Main Function */
int main() {
    int choice;

    do {
        printf("\n--- SINGLY LINKED LIST MENU ---");
        printf("\n1. Create Linked List");
        printf("\n2. Delete First Element");
        printf("\n3. Delete Specified Element");
        printf("\n4. Delete Last Element");
        printf("\n5. Display");
        printf("\n6. Exit");
```

C:\Users\HP\Documents\56.c | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 168, Col 1, Pos 3703 | Insert | Read/Write | default


56.c - Code::Blocks 25.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Resources
Start here
cj.c
56.c
Logs & others
C:\Users\HP\Documents\56.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 168, Col 1, Pos 3703
Insert
Read/Write
default

```c
}

/* Main Function */
int main() {
    int choice;

    do {
        printf("\n--- SINGLY LINKED LIST MENU ---");
        printf("\n1. Create Linked List");
        printf("\n2. Delete First Element");
        printf("\n3. Delete Specified Element");
        printf("\n4. Delete Last Element");
        printf("\n5. Display");
        printf("\n6. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: create();
                    break;
            case 2: delete_first();
                    break;
            case 3: delete_specified();
                    break;
            case 4: delete_last();
                    break;
            case 5: display();
                    break;
            case 6: printf("Exiting program.\n");
                    break;
            default: printf("Invalid choice!\n");
        }
    } while (choice != 6);

    return 0;
}
```



```
--- SINGLY LINKED LIST MENU ---
1. Create Linked List
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display
6. Exit
Enter your choice: 1
Enter number of nodes: 2
Enter data: 2
Enter data: 4

--- SINGLY LINKED LIST MENU ---
1. Create Linked List
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display
6. Exit
Enter your choice: 2
Deleted element: 2

--- SINGLY LINKED LIST MENU ---
1. Create Linked List
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display
6. Exit
Enter your choice: 5
Linked list contents:
4 -> NULL

--- SINGLY LINKED LIST MENU ---
1. Create Linked List
2. Delete First Element
3. Delete Specified Element
4. Delete Last Element
5. Display
6. Exit
Enter your choice:
```