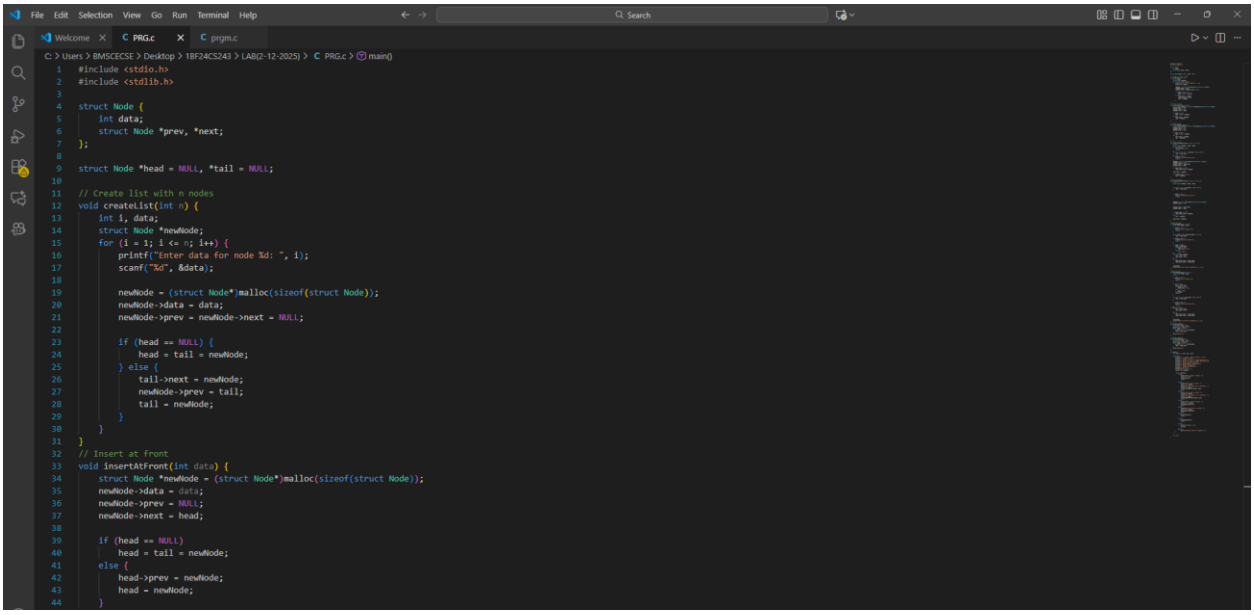
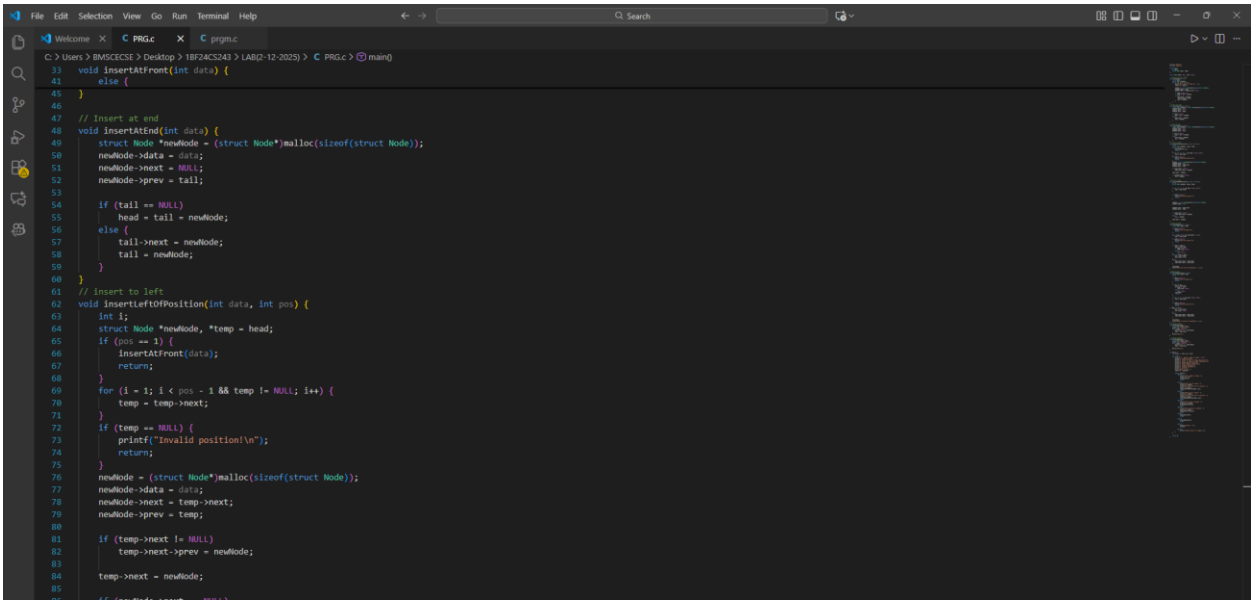
LAB(2-12-2025)

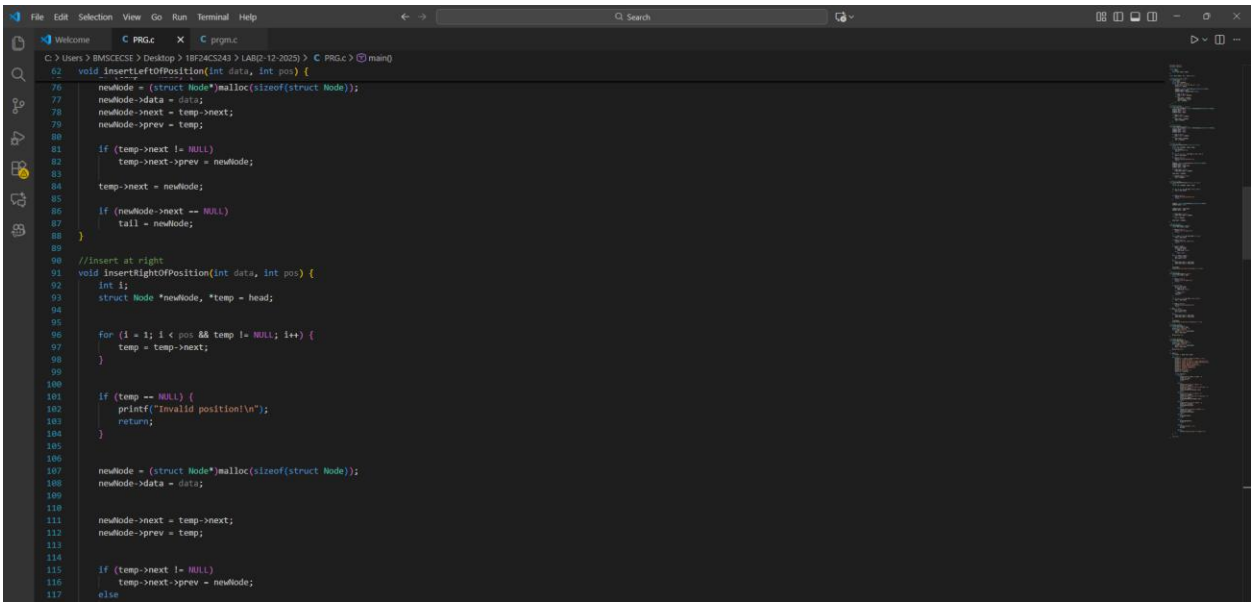
INPUT



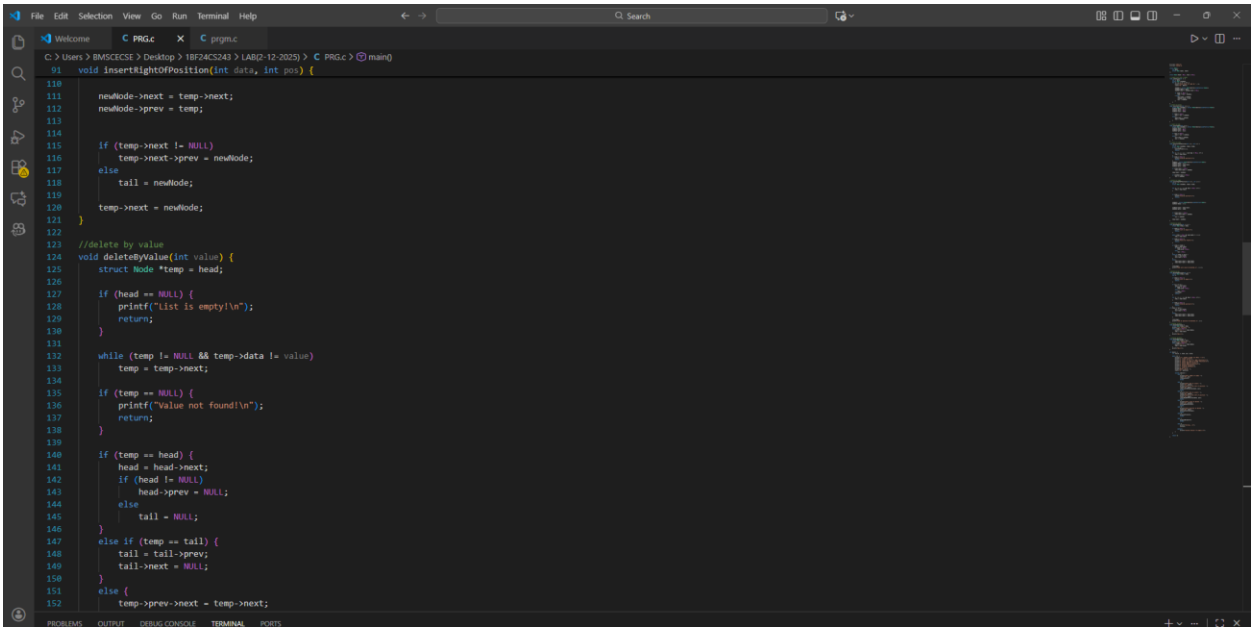
```c
#include <stdio.h>
#include <stdlib.h>

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

struct Node *head = NULL, *tail = NULL;

// Create list with n nodes
void createList(int n) {
    int i, data;
    struct Node *newNode;
    for (i = 1; i <= n; i++) {
        printf("Enter data for node %d: ", i);
        scanf("%d", &data);

        newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = data;
        newNode->prev = newNode->next = NULL;

        if (head == NULL) {
            head = tail = newNode;
        } else {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
    }
}
// Insert at front
void insertAtFront(int data) {
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = head;

    if (head == NULL)
        head = tail = newNode;
    else {
        head->prev = newNode;
        head = newNode;
    }
```



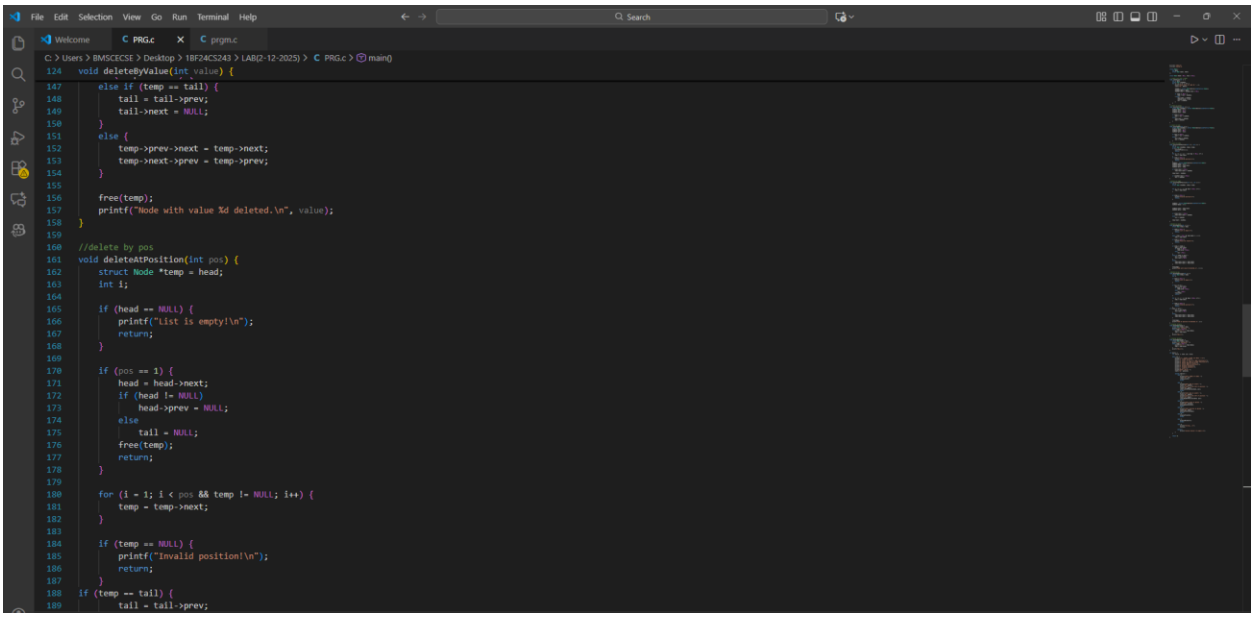
```c
void insertAtFront(int data) {
    else {
}

// Insert at end
void insertAtEnd(int data) {
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = tail;

    if (tail == NULL)
        head = tail = newNode;
    else {
        tail->next = newNode;
        tail = newNode;
    }
}
// insert to left
void insertLeftOfPosition(int data, int pos) {
    int i;
    struct Node *newNode, *temp = head;
    if (pos == 1) {
        insertAtFront(data);
        return;
    }
    for (i = 1; i < pos - 1 && temp != NULL; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Invalid position!\n");
        return;
    }
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = temp->next;
    newNode->prev = temp;

    if (temp->next != NULL)
        temp->next->prev = newNode;

    temp->next = newNode;
```



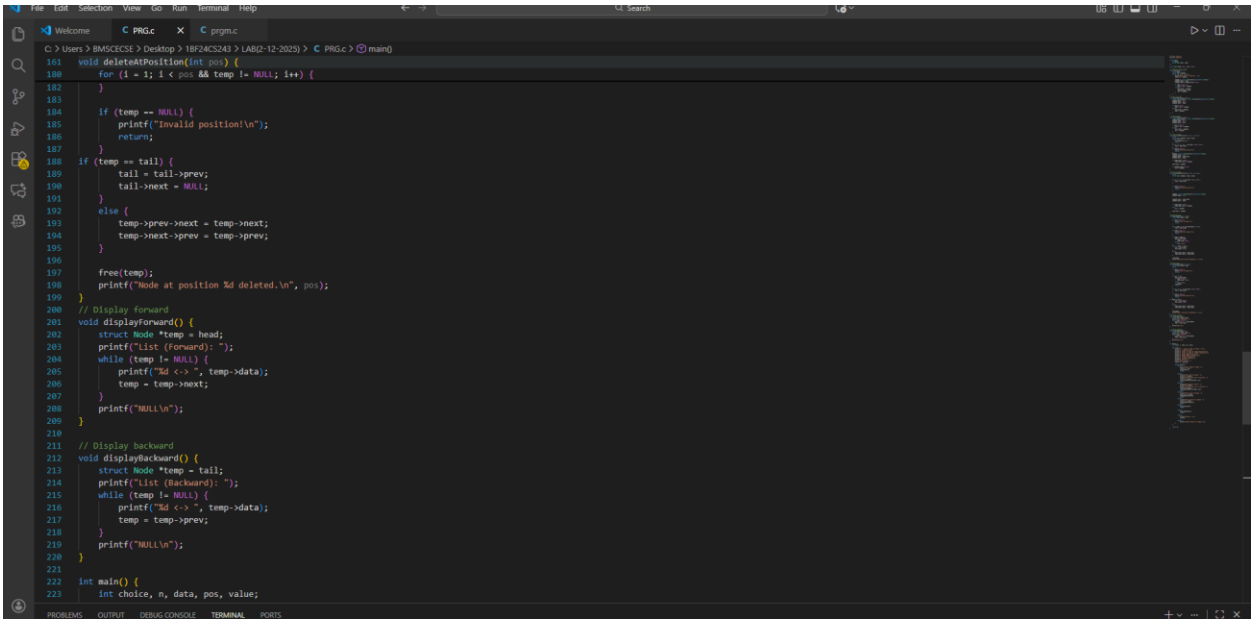
```
void insertLeftOfPosition(int data, int pos) {
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = temp->next;
    newNode->prev = temp;

    if (temp->next != NULL)
        temp->next->prev = newNode;

    temp->next = newNode;

    if (newNode->next == NULL)
        tail = newNode;
}

//insert at right
void insertRightOfPosition(int data, int pos) {
    int i;
    struct Node *newNode, *temp = head;


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


    if (temp == NULL) {
        printf("Invalid position!\n");
        return;
    }


    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;


    newNode->next = temp->next;
    newNode->prev = temp;


    if (temp->next != NULL)
        temp->next->prev = newNode;
    else
```



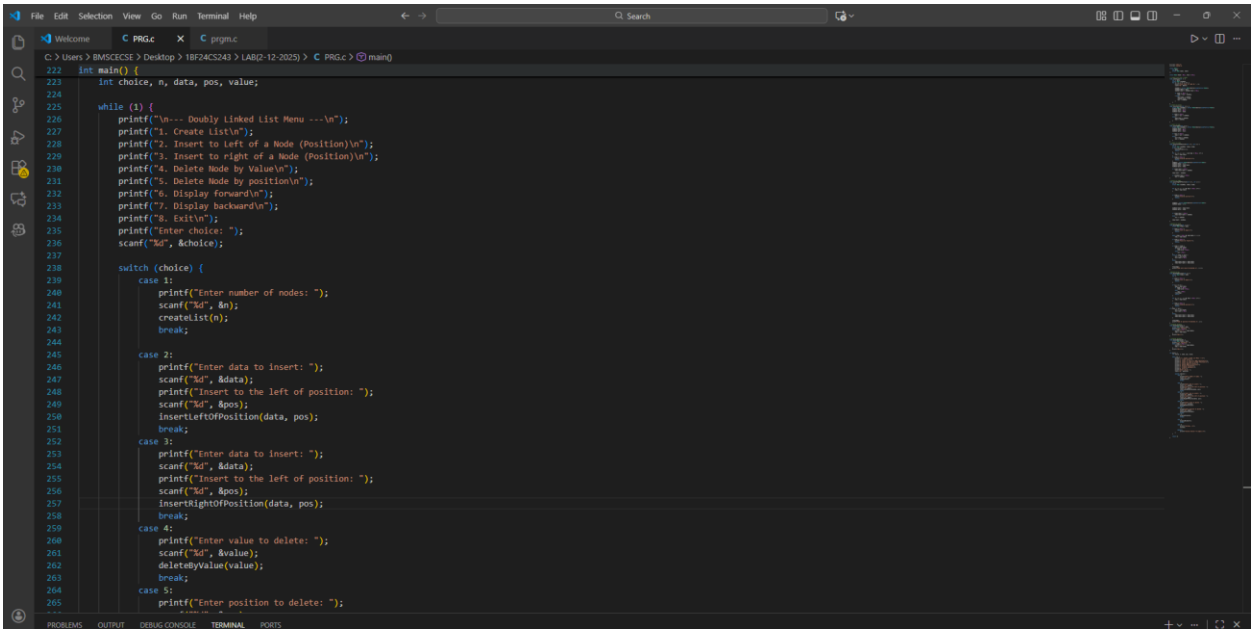
```
void insertRightOfPosition(int data, int pos) {

    newNode->next = temp->next;
    newNode->prev = temp;


    if (temp->next != NULL)
        temp->next->prev = newNode;
    else
        tail = newNode;

    temp->next = newNode;
}

//delete by value
void deleteByValue(int value) {
    struct Node *temp = head;

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

    while (temp != NULL && temp->data != value)
        temp = temp->next;

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

    if (temp == head) {
        head = head->next;
        if (head != NULL)
            head->prev = NULL;
        else
            tail = NULL;
    }
    else if (temp == tail) {
        tail = tail->prev;
        tail->next = NULL;
    }
    else {
        temp->prev->next = temp->next;
```



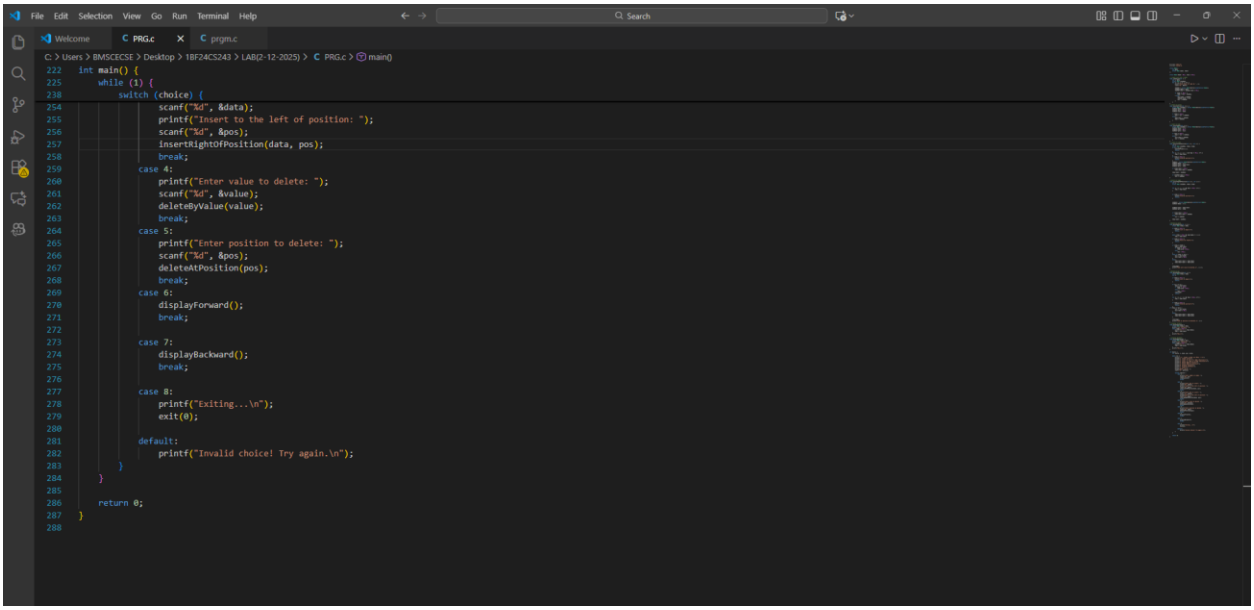
```
void deleteByValue(int value) {
    else if (temp == tail) {
        tail = tail->prev;
        tail->next = NULL;
    }
    else {
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
    }

    free(temp);
    printf("Node with value %d deleted.\n", value);
}

//delete by pos
void deleteAtPosition(int pos) {
    struct Node *temp = head;
    int i;

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

    if (pos == 1) {
        head = head->next;
        if (head != NULL)
            head->prev = NULL;
        else
            tail = NULL;
        free(temp);
        return;
    }

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

    if (temp == NULL) {
        printf("Invalid position!\n");
        return;
    }
if (temp == tail) {
        tail = tail->prev;
```



```
void deleteAtPosition(int pos) {
    for (i = 1; i < pos && temp != NULL; i++) {
    }

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

    free(temp);
    printf("Node at position %d deleted.\n", pos);
}
// Display forward
void displayForward() {
    struct Node *temp = head;
    printf("List (Forward): ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

// Display backward
void displayBackward() {
    struct Node *temp = tail;
    printf("List (Backward): ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->prev;
    }
    printf("NULL\n");
}

int main() {
    int choice, n, data, pos, value;
```

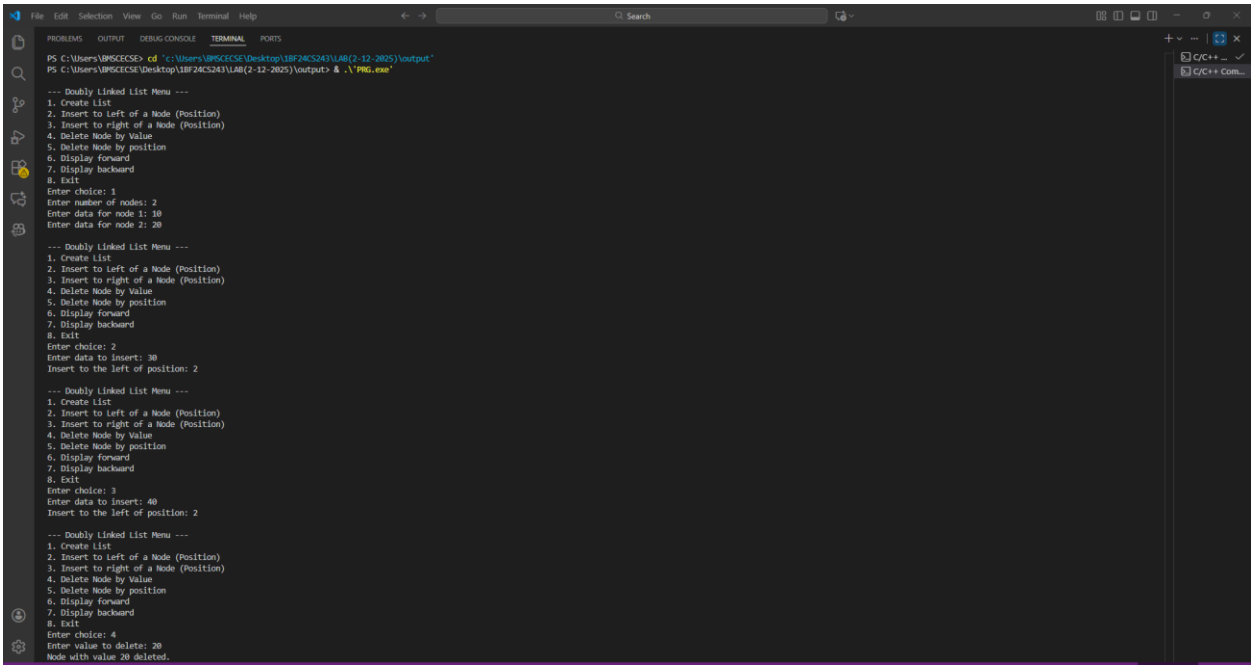


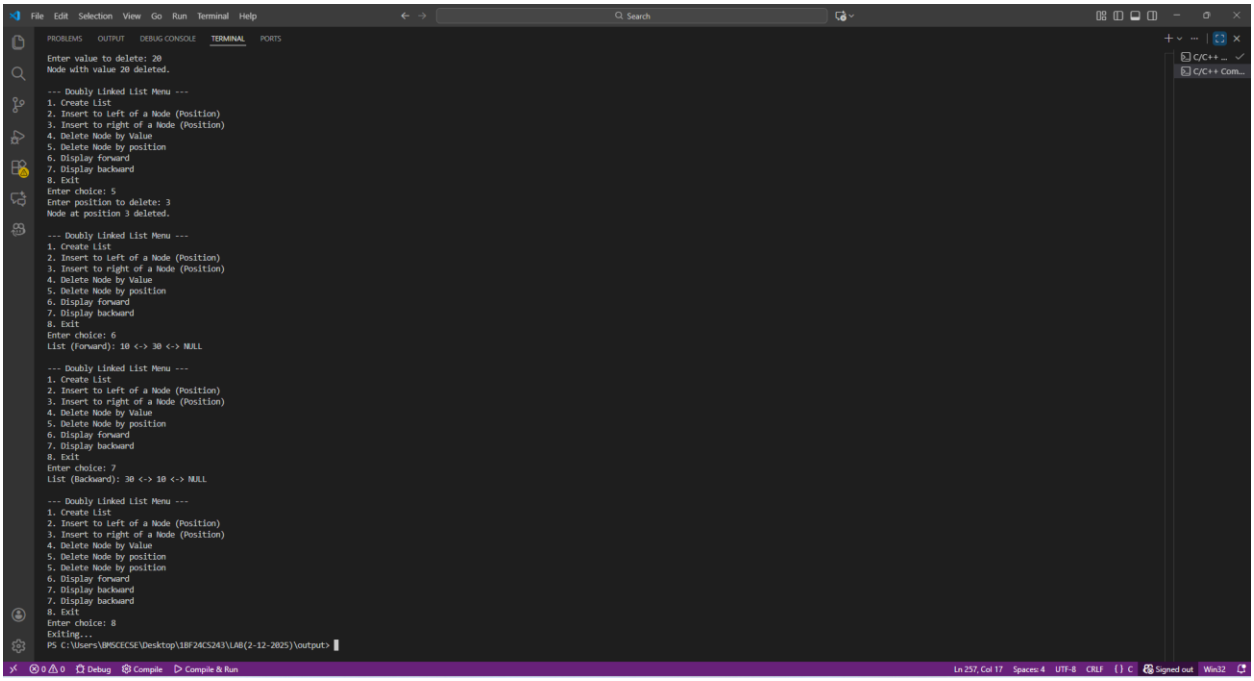
```
int main() {
    int choice, n, data, pos, value;

    while (1) {
        printf("\n--- Doubly Linked List Menu ---\n");
        printf("1. Create List\n");
        printf("2. Insert to Left of a Node (Position)\n");
        printf("3. Insert to right of a Node (Position)\n");
        printf("4. Delete Node by Value\n");
        printf("5. Delete Node by position\n");
        printf("6. Display forward\n");
        printf("7. Display backward\n");
        printf("8. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

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

            case 2:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                printf("Insert to the left of position: ");
                scanf("%d", &pos);
                insertLeftOfPosition(data, pos);
                break;
            case 3:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                printf("Insert to the left of position: ");
                scanf("%d", &pos);
                insertRightOfPosition(data, pos);
                break;
            case 4:
                printf("Enter value to delete: ");
                scanf("%d", &value);
                deleteByValue(value);
                break;
            case 5:
                printf("Enter position to delete: ");
```



```
int main() {
    while (1) {
        switch (choice) {
                scanf("%d", &data);
                printf("Insert to the left of position: ");
                scanf("%d", &pos);
                insertRightOfPosition(data, pos);
                break;
            case 4:
                printf("Enter value to delete: ");
                scanf("%d", &value);
                deleteByValue(value);
                break;
            case 5:
                printf("Enter position to delete: ");
                scanf("%d", &pos);
                deleteAtPosition(pos);
                break;
            case 6:
                displayForward();
                break;

            case 7:
                displayBackward();
                break;

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

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

    return 0;
}
```

OUTPUT


File Edit Selection View Go Run Terminal Help
Search
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\BMSCECSE> cd 'c:\Users\BMSCECSE\Desktop\1BF24CS243\LAB(2-12-2025)\output'
PS C:\Users\BMSCECSE\Desktop\1BF24CS243\LAB(2-12-2025)\output> & .\'PRG.exe'
--- Doubly Linked List Menu ---
1. Create List
2. Insert to Left of a Node (Position)
3. Insert to right of a Node (Position)
4. Delete Node by Value
5. Delete Node by position
6. Display forward
7. Display backward
8. Exit
Enter choice: 1
Enter number of nodes: 2
Enter data for node 1: 10
Enter data for node 2: 20
--- Doubly Linked List Menu ---
1. Create List
2. Insert to Left of a Node (Position)
3. Insert to right of a Node (Position)
4. Delete Node by Value
5. Delete Node by position
6. Display forward
7. Display backward
8. Exit
Enter choice: 2
Enter data to insert: 30
Insert to the left of position: 2
--- Doubly Linked List Menu ---
1. Create List
2. Insert to Left of a Node (Position)
3. Insert to right of a Node (Position)
4. Delete Node by Value
5. Delete Node by position
6. Display forward
7. Display backward
8. Exit
Enter choice: 3
Enter data to insert: 40
Insert to the left of position: 2
--- Doubly Linked List Menu ---
1. Create List
2. Insert to Left of a Node (Position)
3. Insert to right of a Node (Position)
4. Delete Node by Value
5. Delete Node by position
6. Display forward
7. Display backward
8. Exit
Enter choice: 4
Enter value to delete: 20
Node with value 20 deleted.
C/C++ ...
C/C++ Com...


File Edit Selection View Go Run Terminal Help
Search
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Enter value to delete: 20
Node with value 20 deleted.
--- Doubly Linked List Menu ---
1. Create List
2. Insert to Left of a Node (Position)
3. Insert to right of a Node (Position)
4. Delete Node by Value
5. Delete Node by position
6. Display forward
7. Display backward
8. Exit
Enter choice: 5
Enter position to delete: 3
Node at position 3 deleted.
--- Doubly Linked List Menu ---
1. Create List
2. Insert to Left of a Node (Position)
3. Insert to right of a Node (Position)
4. Delete Node by Value
5. Delete Node by position
6. Display forward
7. Display backward
8. Exit
Enter choice: 6
List (Forward): 10 <-> 30 <-> NULL
--- Doubly Linked List Menu ---
1. Create List
2. Insert to Left of a Node (Position)
3. Insert to right of a Node (Position)
4. Delete Node by Value
5. Delete Node by position
6. Display forward
7. Display backward
8. Exit
Enter choice: 7
List (Backward): 30 <-> 10 <-> NULL
--- Doubly Linked List Menu ---
1. Create List
2. Insert to Left of a Node (Position)
3. Insert to right of a Node (Position)
4. Delete Node by Value
5. Delete Node by position
5. Delete Node by position
6. Display forward
7. Display backward
7. Display backward
8. Exit
Enter choice: 8
Exiting...
PS C:\Users\BMSCECSE\Desktop\1BF24CS243\LAB(2-12-2025)\output>
C/C++ ...
C/C++ Com...
Debug
Compile
Compile & Run
Ln 257, Col 17
Spaces: 4
UTF-8
CRLF
C
Signed out
Win32