Program 7

Write A Program to Implement doubly link list with primitive operations

a) Create a doubly linked list.

b) Insert a new node to the left of the node.

c) Delete the node based on a specific value

d) Display the contents of the list

Code:

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

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

Node* createNode(int data);
void insertAtBeginning(Node** head, int data);
void insertAtEnd(Node** head, int data);
void insertAtPosition(Node** head, int data, int position);
void displayList(Node* head);

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

    while (1) {
        printf("\nDoubly Linked List Operations:\n");
        printf("1. Insert at the beginning\n");
        printf("2. Insert at a specific position\n");
printf("3. Insert at the end\n");
printf("4. Display list\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
  case 1:
    printf("Enter data to insert at the beginning: ");
    scanf("%d", &data);
    insertAtBeginning(&head, data);
    break;

  case 2:
    printf("Enter data to insert: ");
    scanf("%d", &data);
    printf("Enter the position: ");
    scanf("%d", &position);
    insertAtPosition(&head, data, position);
    break;

  case 3:
    printf("Enter data to insert at the end: ");
    scanf("%d", &data);
    insertAtEnd(&head, data);
    break;

  case 4:
    displayList(head);
    break;

  case 5:
```

```
                printf("Exiting program.\n");
                exit(0);

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

    return 0;
}

Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

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

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

void insertAtPosition(Node** head, int data, int position) {
    if (position < 1) {
        printf("Invalid position! Position should be 1 or greater.\n");
        return;
    }

    if (position == 1) {
        insertAtBeginning(head, data);
        return;
    }

    Node* newNode = createNode(data);
    Node* temp = *head;
    int count = 1;

    while (temp != NULL && count < position - 1) {
        temp = temp->next;
        count++;
    }

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

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

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

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

void displayList(Node* head) {
    if (head == NULL) {
        printf("The list is empty.\n");
        return;
    }

    Node* temp = head;
    printf("Doubly Linked List: ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
```