Lab 13.1

Program:

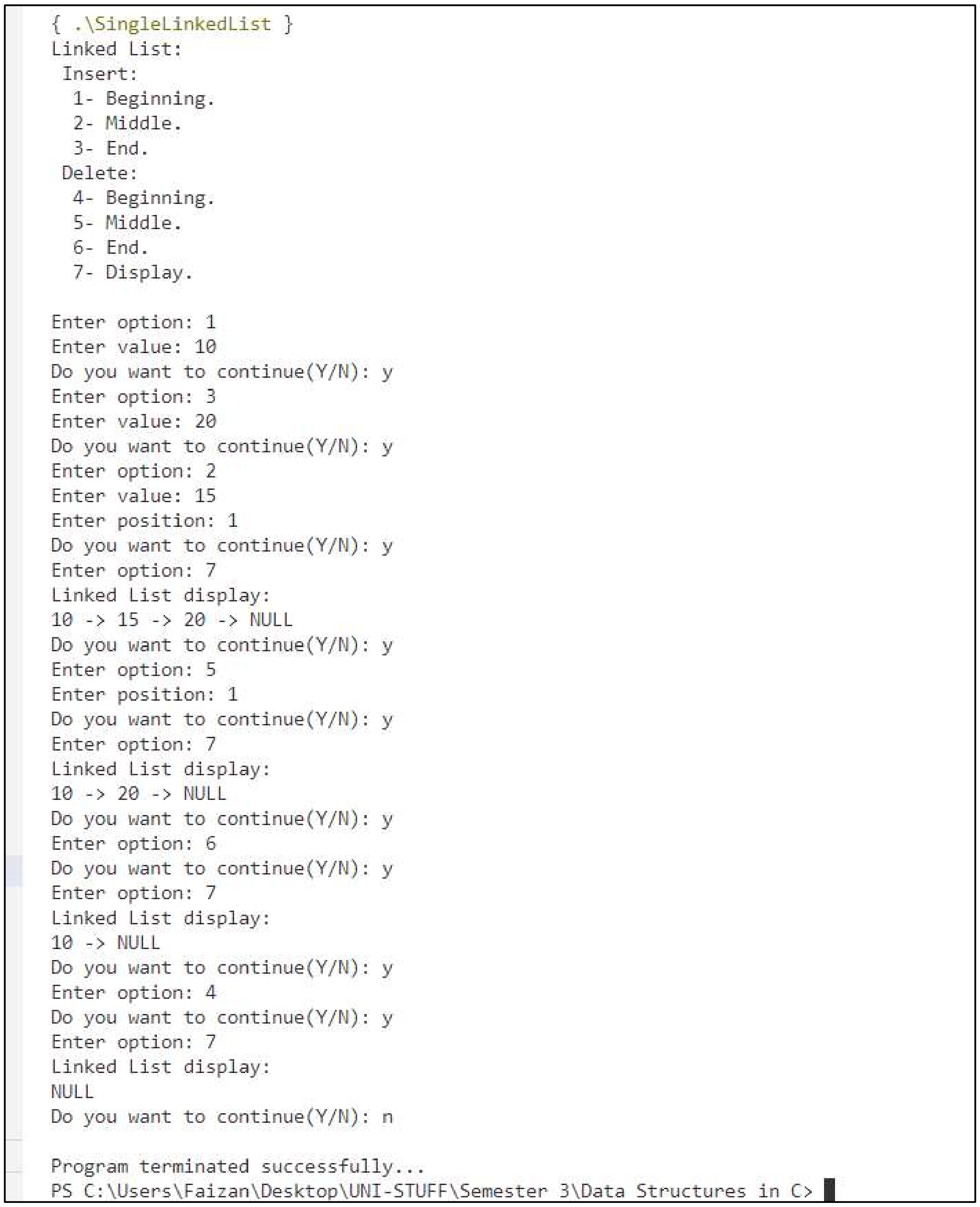
Write a program to perform Insertion and Deletion from the beginning, middle, and the end of a Singly Linked List.

Source code:

```c
#include<stdio.h>
#include<stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
Node* createNode(int data) {
    Node* NN = (Node*)malloc(sizeof(Node));
    if(NN == NULL) {
        printf("List overflow...\n");
        exit(1);
    }
    NN->data = data;
    NN->next = NULL;
    return NN;
}

void insert_beginning(Node** start, int data) {
    Node* NN = createNode(data);
    NN->next = *start;
    *start = NN;
}

void insert_middle(Node** start, int data, int position) {
    if(position < 0){
        printf("Invalid position...\n");
        return;
    }
    if(position == 0){
        insert_beginning(start, data);
        return;
    }
    Node* NN = createNode(data);
    Node* temp = *start;
    for(int i = 0; temp != NULL && i < (position-1); i++)
    {
        temp = temp->next;
    }
    if(temp == NULL) {
        printf("Incorrect position...\n");
```

```
            free(NN);
            return;
        }
        NN->next = temp->next;
        temp->next = NN;
    }

    void insert_end(Node** start, int data) {
        Node* NN = createNode(data);
        if(*start == NULL) {
            *start = NN;
            return;
        }
        Node* end = *start;
        while(end->next != NULL) {
            end = end->next;
        }
        end->next = NN;
    }

    void delete_beginning(Node** start) {
        if(*start == NULL){
            printf("List underflow.\n");
            return;
        }
        Node* temp = *start;
        *start = temp->next;
        free(temp);
    }

    void delete_middle(Node** start, int position) {
        if(*start == NULL){
            printf("List underflow.\n");
            return;
        }
        if(position == 0){
            delete_beginning(start);
            return;
        }
        Node* temp = *start;
        for (int i = 0; temp != NULL && i < position - 1; i++) {
            temp = temp->next;
        }
        if (temp == NULL || temp->next == NULL) {
            printf("Incorrect position.\n");
            return;
        }
        Node* NN = temp->next;
    temp->next = NN->next;
    free(NN);
}

void delete_end(Node** start) {
    if(*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    if((*start)->next == NULL) {
        free(*start);
        *start = NULL;
        return;
    }
    Node* temp = *start;
    while((temp->next)->next != NULL) {
        temp = temp->next;
    }
    free(temp->next);
    temp->next = NULL;
}

void display(Node* node) {
    while (node != NULL) {
        printf("%d -> ", node->data);
        node = node->next;
    }
    printf("NULL\n");
}

int main()
{
    Node* start = NULL;
    int val, position, optn;
    char contn;

    printf("Linked List:\n Insert:\n  1- Beginning.\n  2- Middle.\n  3- End.\n 
Delete:\n  4- Beginning.\n  5- Middle.\n  6- End.\n  7- Display.\n\n");

    do {
        printf("Enter option: ");
        scanf("%d", &optn);

        switch(optn){
            case 1:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_beginning(&start, val);
                break;
            case 2:
                printf("Enter value: ");
                scanf("%d", &val);
                printf("Enter position: ");
                scanf("%d", &position);
                insert_middle(&start, val, position);
                break;
            case 3:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_end(&start, val);
                break;
            case 4:
                delete_beginning(&start);
                break;
            case 5:
                printf("Enter position: ");
                scanf("%d", &position);
                delete_middle(&start, position);
                break;
            case 6:
                delete_end(&start);
                break;
            case 7:
                printf("Linked List display:\n");
                display(start);
                break;
            default:
                printf("Incorrect option...\n");
                break;
        }
        printf("Do you want to continue(Y/N): ");
        scanf(" %c", &contn);
    }
    while(contn == 'Y' || contn == 'y');

    printf("\nProgram terminated successfully...\n");

    return 0;
}
```

Output:



```
{ .\SingleLinkedList }
Linked List:
 Insert:
  1- Beginning.
  2- Middle.
  3- End.
 Delete:
  4- Beginning.
  5- Middle.
  6- End.
  7- Display.

Enter option: 1
Enter value: 10
Do you want to continue(Y/N): y
Enter option: 3
Enter value: 20
Do you want to continue(Y/N): y
Enter option: 2
Enter value: 15
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
10 -> 15 -> 20 -> NULL
Do you want to continue(Y/N): y
Enter option: 5
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
10 -> 20 -> NULL
Do you want to continue(Y/N): y
Enter option: 6
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
10 -> NULL
Do you want to continue(Y/N): y
Enter option: 4
Do you want to continue(Y/N): y
Enter option: 7
Linked List display:
NULL
Do you want to continue(Y/N): n

Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```

Lab 13.2

Program:

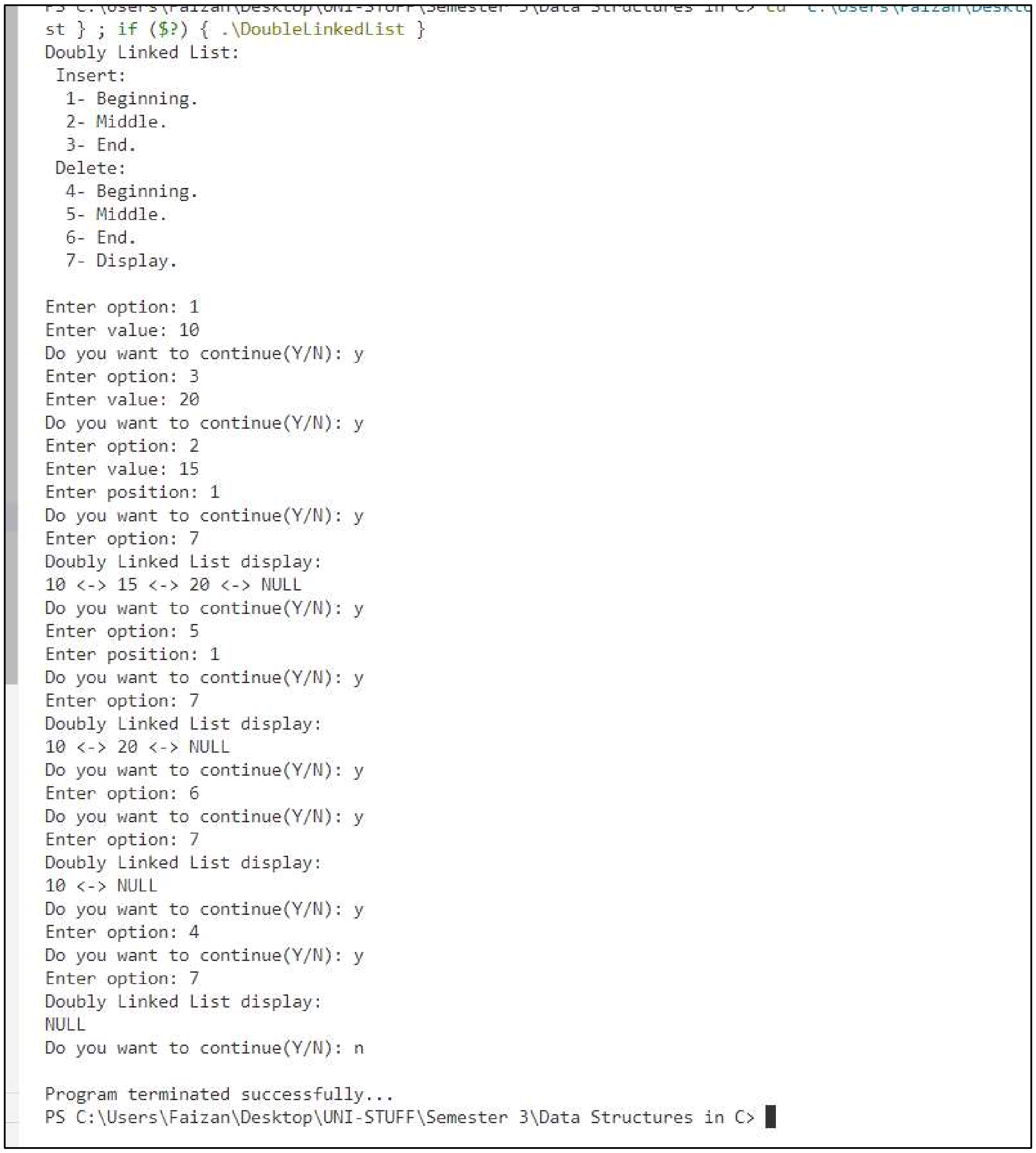
Write a program to perform Insertion and Deletion from the beginning, middle, and the end of a Doubly Linked List.

Source code:

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;
Node* createNode(int data) {
    Node* NN = (Node*)malloc(sizeof(Node));
    if (NN == NULL) {
        printf("List overflow...\n");
        exit(1);
    }
    NN->data = data;
    NN->next = NULL;
    NN->prev = NULL;
    return NN;
}
void insert_beginning(Node** start, int data) {
    Node* NN = createNode(data);
    NN->next = *start;
    if (*start != NULL) {
        (*start)->prev = NN;
    }
    *start = NN;
}
void insert_middle(Node** start, int data, int position) {
    if (position < 0) {
        printf("Invalid position...\n");
        return;
    }
    if (position == 0) {
        insert_beginning(start, data);
        return;
    }
    Node* NN = createNode(data);
    Node* temp = *start;
    for (int i = 0; temp != NULL && i < (position - 1); i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Incorrect position...\n");
        free(NN);
        return;
    }
    NN->next = temp->next;
    NN->prev = temp;
    if (temp->next != NULL) {
        temp->next->prev = NN;
    }
    temp->next = NN;
}

void insert_end(Node** start, int data) {
    Node* NN = createNode(data);
    if (*start == NULL) {
        *start = NN;
        return;
    }
    Node* end = *start;
    while (end->next != NULL) {
        end = end->next;
    }
    end->next = NN;
    NN->prev = end;
}

void delete_beginning(Node** start) {
    if (*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    Node* temp = *start;
    *start = temp->next;
    if (*start != NULL) {
        (*start)->prev = NULL;
    }
    free(temp);
}

void delete_middle(Node** start, int position) {
    if (*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    if (position == 0) {
        delete_beginning(start);
        return;
    }
    Node* temp = *start;
    for (int i = 0; temp != NULL && i < position; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Incorrect position.\n");
        return;
    }
    if (temp->prev != NULL) {
        temp->prev->next = temp->next;
    }
    if (temp->next != NULL) {
        temp->next->prev = temp->prev;
    }
    free(temp);
}
void delete_end(Node** start) {
    if (*start == NULL) {
        printf("List underflow.\n");
        return;
    }
    if ((*start)->next == NULL) {
        free(*start);
        *start = NULL;
        return;
    }
    Node* temp = *start;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->prev->next = NULL;
    free(temp);
}
void display(Node* node) {
    while (node != NULL) {
        printf("%d <-> ", node->data);
        node = node->next;
    }
    printf("NULL\n");
}
int main() {
    Node* start = NULL;
    int val, position, optn;
    char contn;

    printf("Doubly Linked List:\n Insert:\n  1- Beginning.\n  2- Middle.\n  3-
End.\n Delete:\n  4- Beginning.\n  5- Middle.\n  6- End.\n  7- Display.\n\n");
```

```
    do {
        printf("Enter option: ");
        scanf("%d", &optn);
        switch (optn) {
            case 1:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_beginning(&start, val);
                break;
            case 2:
                printf("Enter value: ");
                scanf("%d", &val);
                printf("Enter position: ");
                scanf("%d", &position);
                insert_middle(&start, val, position);
                break;
            case 3:
                printf("Enter value: ");
                scanf("%d", &val);
                insert_end(&start, val);
                break;
            case 4:
                delete_beginning(&start);
                break;
            case 5:
                printf("Enter position: ");
                scanf("%d", &position);
                delete_middle(&start, position);
                break;
            case 6:
                delete_end(&start);
                break;
            case 7:
                printf("Doubly Linked List display:\n");
                display(start);
                break;
            default:
                printf("Incorrect option...\n");
                break;
        }
        printf("Do you want to continue(Y/N): ");
        scanf(" %c", &contn);
    } while (contn == 'Y' || contn == 'y');
    printf("\nProgram terminated successfully...\n");
    return 0;
}
```

Output:



```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C> cd "c:\Users\Faizan\Deskto
st } ; if ($?) { .\DoubleLinkedList }
Doubly Linked List:
 Insert:
  1- Beginning.
  2- Middle.
  3- End.
 Delete:
  4- Beginning.
  5- Middle.
  6- End.
  7- Display.

Enter option: 1
Enter value: 10
Do you want to continue(Y/N): y
Enter option: 3
Enter value: 20
Do you want to continue(Y/N): y
Enter option: 2
Enter value: 15
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Doubly Linked List display:
10 <-> 15 <-> 20 <-> NULL
Do you want to continue(Y/N): y
Enter option: 5
Enter position: 1
Do you want to continue(Y/N): y
Enter option: 7
Doubly Linked List display:
10 <-> 20 <-> NULL
Do you want to continue(Y/N): y
Enter option: 6
Do you want to continue(Y/N): y
Enter option: 7
Doubly Linked List display:
10 <-> NULL
Do you want to continue(Y/N): y
Enter option: 4
Do you want to continue(Y/N): y
Enter option: 7
Doubly Linked List display:
NULL
Do you want to continue(Y/N): n

Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\Data Structures in C>
```