*/\*\**

*\* C program to create and display Doubly linked list*

*\*/*

#include <stdio.h>

#include <stdlib.h>

*/\**

*\* Basic structure of Node*

*\*/*

struct node {

    int data;

    struct node \* prev;

    struct node \* next;

}\*head, \*last;

*/\**

*\* Function used in this program*

*\*/*

void createList(int n);

void displayList();

void insertAtBeginning(int data);

void insertAtEnd(int data);

void insertAfter(int data , int k);

void insert\_Before(int x, int value);

void delete\_At\_Front();

void delete\_At\_End();

void delete\_After(int x);

void delete\_Before(int x);

int main()

{

    int n, choice,x , y;

    choice = 1;

    head = NULL;

    last = NULL;

    printf("============================================\n");

    printf("DOUBLY LINKED LIST PROGRAM\n");

    printf("============================================\n");

    printf("Enter the number of nodes you want to create: ");

    scanf("%d", &n);

*// Create doubly linked list*

    createList(n);

*// Display the doubly linked list*

    displayList();

*/\**

*\* Run forever until user chooses 0*

*\*/*

    while(choice != 0)

    {

*/\**

*\* Menu creation to use the program*

*\*/*

        printf("============================================\n");

        printf("DOUBLY LINKED LIST PROGRAM\n");

        printf("============================================\n");

        printf("1. Insert node - at beginning\n");

        printf("2. Insert node - at end\n");

        printf("3. Insert node - after a node\n");

        printf("4. Insert node - before a node\n");

        printf("5. Delete node - at beginning\n");

        printf("6. Delete node - at end\n");

        printf("7. Delete node - after a node\n");

        printf("8. Delete node - before a node\n");

        printf("9. Display list\n");

        printf("0. Exit\n");

        printf("--------------------------------------------\n");

        printf("Enter your choice : ");

        scanf("%d", &choice);

*/\**

*\* Choose from different menu operation*

*\*/*

        switch(choice)

        {

            case 1:

                printf("Enter the data to be added at first : ");

                scanf("%d", &x);

*// insert*

                insertAtBeginning(x);

*// Display*

                displayList();

                break;

            case 2:

                printf("Enter the data to be added at last : ");

                scanf("%d", &x);

*// insert*

                insertAtEnd(x);

*// Display*

                displayList();

            case 3:

                printf("Enter the data to be added after a node : ");

                scanf("%d", &x);

                printf("After which node you want to put the above data : ");

                scanf("%d", &y);

*// insert*

                insertAfter(x , y);

*// Display*

                displayList();

                break;

            case 4:

*// insert before given data*

                printf("Enter the value to be insterted ");

                scanf("%d" , &x);

                printf("Enter the value of the node before which you want to insert the new node");

                scanf("%d" , &y);

                insert\_Before(y , x);

                displayList();

                break;

            case 5:

*// delete from first*

                delete\_At\_Front();

                displayList();

                break;

            case 6:

*// delete from end*

                delete\_At\_End();

                displayList();

                break;

            case 7:

*// delete after*

                printf("Enter the value of the node whose next node you want to delete ");

                scanf("%d" , &x);

                delete\_After(x);

                displayList();

                break;

            case 8:

*// delete before*

                printf("Enter the value of the node whose previous node you want to delete ");

                scanf("%d" , &x);

                delete\_Before(x);

                displayList();

                break;

            case 0:

                break;

            default:

                printf("Error! Invalid choice. Please choose between 0-5");

        }

        printf("\n");

    }

    return 0;

}

*/\*\**

*\* Create a doubly linked list of n nodes.*

*\* @n Number of nodes to be created*

*\*/*

void createList(int n)

{

    int i, data;

    struct node \*newNode;

    if(n >= 1)

    {

        head = (struct node \*)malloc(sizeof(struct node));

        if(head != NULL)

        {

            printf("Enter data of 1 node: ");

            scanf("%d", &data);

            head->data = data;

            head->prev = NULL;

            head->next = NULL;

            last = head;

*/\**

*\* Create rest of the n-1 nodes*

*\*/*

            for(i=2; i<=n; i++)

            {

                newNode = (struct node \*)malloc(sizeof(struct node));

                if(newNode != NULL)

                {

                    printf("Enter data of %d node: ", i);

                    scanf("%d", &data);

                    newNode->data = data;

                    newNode->prev = last; *// Link new node with the previous node*

                    newNode->next = NULL;

                    last->next = newNode; *// Link previous node with the new node*

                    last = newNode;          *// Make new node as last/previous node*

                }

                else

                {

                    printf("Unable to allocate memory.");

                    break;

                }

            }

            printf("\nDOUBLY LINKED LIST CREATED SUCCESSFULLY\n");

        }

        else

        {

            printf("Unable to allocate memory");

        }

    }

}

*/\*\**

*\* Displays the content of the list from beginning to end*

*\*/*

void displayList()

{

    struct node \* temp;

    int n = 1;

    if(head == NULL)

    {

        printf("List is empty.");

    }

    else

    {

        temp = head;

        printf("\n\nYOUR DOUBLY LINKED LIST:\n");

        while(temp != NULL)

        {

            printf("DATA of %d node = %d\n", n, temp->data);

            n++;

*/\* Move the current pointer to next node \*/*

            temp = temp->next;

        }

    }

}

void insertAtBeginning(int data)

{

    struct node \* newNode;

    if(head == NULL)

    {

        printf("Error, List is Empty!\n");

    }

    else

    {

        newNode = (struct node \*)malloc(sizeof(struct node));

        newNode->data = data;

        newNode->next = head; *// Point to next node which is currently head*

        newNode->prev = NULL; *// Previous node of first node is NULL*

*/\* Link previous address field of head with newnode \*/*

        head->prev = newNode;

*/\* Make the new node as head node \*/*

        head = newNode;

        printf("NODE INSERTED SUCCESSFULLY AT THE BEGINNING OF THE LIST\n");

    }

}

void insertAtEnd(int data)

{

    struct node \* newNode;

    if(last == NULL)

    {

        printf("Error, List is empty!\n");

    }

    else

    {

        newNode = (struct node \*)malloc(sizeof(struct node));

        newNode->data = data;

        newNode->next = NULL;

        newNode->prev = last;

        last->next = newNode;

        last = newNode;

        printf("NODE INSERTED SUCCESSFULLY AT THE END OF LIST\n");

    }

}

void insertAfter(int data , int k){

*//struct node \* newNode;*

    if(head == NULL)

    {

        printf("Error, List is Empty!\n");

    }

    else

    {

        struct node \*p;

        p = malloc(sizeof(struct node));

        p->data = data;

        struct node \*temp;

        temp = head;

        while(temp->data != k){

            temp = temp->next;

        }

        struct node \*a;

        a = temp->next;

        temp->next = p;

        p->next = a;

        a->prev = p;

        p->prev = temp;

        printf("NODE INSERTED SUCCESSFULLY AT THE BEGINNING OF THE LIST\n");

    }

}

void insert\_Before(int x, int value)

{

    struct node \*p;

    p = malloc(sizeof(struct node));

    p->data = value;

    struct node \*temp;

    temp = head;

    while(temp->data != x){

        temp = temp->next;

    }

    struct node\* a;

    a = temp -> prev;

    a->next = p;

    p->prev = a;

    temp->prev = p;

    p->next = temp;

    printf("NODE INSERTED SUCCESSFULLY AT THE BEGINNING OF THE LIST\n");

}

*/\**

*\* delete from first*

*\*/*

void delete\_At\_Front()

{

    struct node \*temp;

    temp = head->next;

    head->next = NULL;

    head = temp;

    head->prev = NULL;

}

*/\**

*\* delete from end*

*\*/*

void delete\_At\_End()

{

    struct node \*temp;

    temp = head;

    while(temp->next->next != NULL){

        temp = temp->next;

    }

    struct node \*a;

    a = temp->next;

    temp->next = NULL;

    a->prev = NULL;

}

*/\**

*\* delete after a node*

*\*/*

void delete\_After(int x)

{

    struct node \*temp;

    temp = head;

    while(temp->data != x){

        temp = temp->next;

    }

    struct node \*a;

    a = temp->next;

    struct node \*b;

    b = a->next;

    temp->next = b;

    b->prev = temp;

}

*/\**

*\* delete before a node*

*\*/*

void delete\_Before(int x){

    struct node \*temp;

    temp = head;

    while(temp->data != x){

        temp = temp->next;

    }

    struct node \*a;

    a = temp->prev;

    struct node \*b;

    b = a->prev;

    temp->prev = b;

    b->next = temp;

}

Output

