*/\*\**

*\* C program to insert a new node in a Circular Linked List*

*\*/*

#include <stdio.h>

#include <stdlib.h>

*/\**

*\* Basic structure of Node*

*\*/*

struct node {

    int data;

    struct node \* next;

}\*head;

*/\**

*\* Functions used in this program*

*\*/*

void createList(int n);

void displayList();

void insertAtBeginning(int data);

*//void insertAtN(int data, int position);*

void insertAtEnd(int value);

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

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, data, choice=1;

    int x , val;

    head = NULL;

*/\**

*\* Runs forever until user chooses 0*

*\*/*

    while(choice != 0)

    {

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

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

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

        printf("1. Create List\n");

        printf("2. Display list\n");

        printf("3. Insert node - at beginning\n");

        printf("4. Insert node - at end\n");

        printf("5. Insert node - after a node\n");

        printf("6. Insert node - before a node\n");

        printf("7. Delete node - at beginning\n");

        printf("8. Delete node - at end\n");

        printf("9. Delete node - after a node\n");

        printf("10. Delete node - before a node\n");

        printf("0. Exit\n");

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

        printf("Enter your choice : ");

        scanf("%d", &choice);

        switch(choice)

        {

            case 1:

                printf("Enter the total number of nodes in list: ");

                scanf("%d", &n);

                createList(n);

                displayList();

                break;

            case 2:

                displayList();

                break;

            case 3:

                printf("Enter data to be inserted at beginning: ");

                scanf("%d", &data);

                insertAtBeginning(data);

                break;

            case 4:

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

                scanf("%d", &x);

*// insert*

                insertAtEnd(x);

*// Display*

                displayList();

                break;

            case 5:

*// insert after given data*

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

                scanf("%d" , &val);

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

                scanf("%d" , &x);

                insert\_After(x , val);

                displayList();

                break;

            case 6:

*// insert before given data*

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

                scanf("%d" , &val);

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

                scanf("%d" , &x);

                insert\_Before(x , val);

                displayList();

                break;

            case 7:

*// delete from first*

                delete\_At\_Front();

                displayList();

                break;

            case 8:

*// delete from end*

                delete\_At\_End();

                displayList();

                break;

            case 9:

*// 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 10:

*// 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 0-4");

        }

        printf("\n\n\n\n\n");

    }

    return 0;

}

*/\*\**

*\* Creates a circular linked list of n nodes.*

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

*\*/*

void createList(int n)

{

    int i, data;

    struct node \*prevNode, \*newNode;

    if(n >= 1)

    {

*/\**

*\* Creates and links the head node*

*\*/*

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

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

        scanf("%d", &data);

        head->data = data;

        head->next = NULL;

        prevNode = head;

*/\**

*\* Creates and links rest of the n-1 nodes*

*\*/*

        for(i=2; i<=n; i++)

        {

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

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

            scanf("%d", &data);

            newNode->data = data;

            newNode->next = NULL;

*//Links the previous node with newly created node*

            prevNode->next = newNode;

*//Moves the previous node ahead*

            prevNode = newNode;

        }

*//Links the last node with first node*

        prevNode->next = head;

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

    }

}

*/\*\**

*\* Displays the content of the list*

*\*/*

void displayList()

{

    struct node \*current;

    int n = 1;

    if(head == NULL)

    {

        printf("List is empty.\n");

    }

    else

    {

        current = head;

        printf("DATA IN THE LIST:\n");

        do {

            printf("Data %d = %d\n", n, current->data);

            current = current->next;

            n++;

        }while(current != head);

    }

}

*/\*\**

*\* Inserts a new node at the beginning of the list*

*\* @data Data of the first node*

*\*/*

void insertAtBeginning(int data)

{

    struct node \*newNode, \*current;

    if(head == NULL)

    {

        printf("List is empty.\n");

    }

    else

    {

*/\**

*\* Creates new node, assign data and links it to head*

*\*/*

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

        newNode->data = data;

        newNode->next = head;

*/\**

*\* Traverses to last node and links last node*

*\* with first node which is new node*

*\*/*

        current = head;

        while(current->next != head)

        {

            current = current->next;

        }

        current->next = newNode;

*/\*Makes new node as head node\*/*

        head = newNode;

        printf("NODE INSERTED SUCCESSFULLY\n");

    }

}

*/\*\**

*\* Inserts a new node at any position in the list*

*\* @data Data of the new node*

*\* @position Position where to insert new node*

*\*/*

void insertAtN(int data, int position)

{

    struct node \*newNode, \*current;

    int i;

    if(head == NULL)

    {

        printf("List is empty.\n");

    }

    else if(position == 1)

    {

        insertAtBeginning(data);

    }

    else

    {

*/\**

*\* Creates new node and assign data to it*

*\*/*

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

        newNode->data = data;

*/\**

*\* Traverse to n-1 node*

*\*/*

        current = head;

        for(i=2; i<=position-1; i++)

        {

            current = current->next;

        }

*/\* Links new node with node ahead of it and previous to it\*/*

        newNode->next = current->next;

        current->next = newNode;

        printf("NODE INSERTED SUCCESSFULLY.\n");

    }

}

void insertAtEnd(int value)

{

    struct node \*p,\*q;

    p=malloc(sizeof(struct node));

    p->data=value;

    p->next=NULL;

    q=head;

    while(q->next!=head)

    {

        q = q->next;

    }

    q->next = p;

    p->next = head;

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

}

*/\**

*\* insert after a node*

*\*/*

void insert\_After(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->next;

    temp->next = p;

    p->next = a;

}

*/\**

*\* insert before a node*

*\*/*

void insert\_Before(int x, int value)

{

    struct node \*p;

    p = malloc(sizeof(struct node));

    p->data = value;

    struct node \*temp , \*beforetemp;

    beforetemp = head;

    while(beforetemp->next->data != x){

        beforetemp = beforetemp->next;

    }

    temp = beforetemp->next;

    beforetemp->next = p;

    p->next = temp;

}

*/\**

*\* delete from first*

*\*/*

void delete\_At\_Front()

{

    struct node \*temp;

    temp = head;

    while(temp->next != head){

        temp = temp->next;

    }

    struct node \*a;

    a = head->next;

    temp->next = a;

    head = a;

}

*/\**

*\* delete from end*

*\*/*

void delete\_At\_End()

{

    struct node \*temp;

    temp = head;

    while(temp->next->next != head){

        temp = temp->next;

    }

    temp->next = head;

}

*/\**

*\* delete after a node*

*\*/*

void delete\_After(int x)

{

    struct node \*temp;

    temp = head;

    while(temp->data != x){

        temp = temp->next;

    }

    temp->next = temp->next->next;

}

*/\**

*\* delete before a node*

*\*/*

void delete\_Before(int x){

    struct node \*temp;

    temp = head;

    while(temp->next->next->data != x){

        temp = temp->next;

    }

    temp->next = temp->next->next;

}

Output

