Qs. Write a C program for representation of singly linked lists (create and display it) and then implementation of relevant operations – add, delete from beginning, end at and after specified locations (i.e., before and after a given node)

Code

*/\*\**

*\* Write a C program for representation of singly linked lists (create and display it) and then*

*\* implementation of relevant operations – add, delete from beginning, end at and after specified*

*\* locations (i.e., before and after a given node)*

*\*/*

#include <stdio.h>

#include <stdlib.h>

*/\* Structure of a node \*/*

struct node {

    int data;

    struct node \*next;

}\*head;

void createList(int n);

void traverseList();

void insert\_At\_Front(int value);

void insert\_At\_End(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);

*/\*\**

*\* main method*

*\*/*

int main()

{

    int n;

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

    scanf("%d", &n);

    createList(n);

    printf("\nData in the list \n");

    traverseList();

    int k;

    int val , x;

    printf("");

    printf("Enter 1 for insert at first\n");

    printf("Enter 2 for insert at end\n");

    printf("Enter 3 for insert after a node\n");

    printf("Enter 4 for insert before a node\n");

    printf("Enter 5 for delete at first\n");

    printf("Enter 6 for delete at end\n");

    printf("Enter 7 for delete after a node\n");

    printf("Enter 8 for delete before a node\n");

    printf("");

    scanf("%d" , &k);

    switch (k)

    {

      case 1:

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

        scanf("%d" , &val);

        insert\_At\_Front(val);

        traverseList();

        break;

      case 2:

*// insert at end*

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

        scanf("%d" , &val);

        insert\_At\_End(val);

        traverseList();

        break;

      case 3:

*// 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);

        traverseList();

        break;

      case 4:

*// 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);

        traverseList();

        break;

      case 5:

*// delete from first*

        delete\_At\_Front();

        traverseList();

        break;

      case 6:

*// delete from end*

        delete\_At\_End();

        traverseList();

        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);

        traverseList();

        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);

        traverseList();

        break;

      default:

          printf("Wrong input");

    }

    return 0;

}

*/\**

*\* Create a list of n nodes*

*\*/*

void createList(int n)

{

    struct node \*newNode, \*temp;

    int data, i;

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

*// Terminate if memory not allocated*

    if(head == NULL)

    {

        printf("Unable to allocate memory.");

        exit(0);

    }

*// Input data of node from the user*

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

    scanf("%d", &data);

    head->data = data; *// Link data field with data*

    head->next = NULL; *// Link address field to NULL*

*// Create n - 1 nodes and add to list*

    temp = head;

    for(i=2; i<=n; i++)

    {

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

*/\* If memory is not allocated for newNode \*/*

        if(newNode == NULL)

        {

            printf("Unable to allocate memory.");

            break;

        }

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

        scanf("%d", &data);

        newNode->data = data; *// Link data field of newNode*

        newNode->next = NULL; *// Make sure new node points to NULL*

        temp->next = newNode; *// Link previous node with newNode*

        temp = temp->next;    *// Make current node as previous node*

    }

}

*/\**

*\* Display entire list*

*\*/*

void traverseList()

{

    struct node \*temp;

*// Return if list is empty*

    if(head == NULL)

    {

        printf("List is empty.");

        return;

    }

    temp = head;

    while(temp != NULL)

    {

        printf("Data = %d\n", temp->data); *// Print data of current node*

        temp = temp->next;                 *// Move to next node*

    }

}

*/\*\**

*\* insert at fornt*

*\*/*

void insert\_At\_Front(int value)

{

    struct node \*p;

    p=malloc(sizeof(struct node));

    p->data=value;

    p->next=head;

    head = p;

}

*/\**

*\* insert at end*

*\*/*

void insert\_At\_End(int value)

{

    struct node \*p,\*q;

    p=malloc(sizeof(struct node));

    p->data=value;

    p->next=NULL;

    q=head;

    while(q->next!=NULL)

    {

        q = q->next;

    }

    q->next = p;

}

*/\**

*\* 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 = temp->next;

    }

    temp = beforetemp->next;

    beforetemp->next = p;

    p->next = temp;

}

*/\**

*\* delete from first*

*\*/*

void delete\_At\_Front()

{

    struct node \*temp;

    temp = head->next;

    head->next = NULL;

    head = temp;

}

*/\**

*\* delete from end*

*\*/*

void delete\_At\_End()

{

    struct node \*temp;

    temp = head;

    while(temp->next->next != NULL){

        temp = temp->next;

    }

    temp->next = NULL;

}

*/\**

*\* 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

