**Circular LinkedList**

// implementation, creation, insertion and deletion

#include<stdio.h>

#include<stdlib.h>

typedef struct Node

{

    int data;

    struct Node\* next;

}Node;

Node\* head=NULL;

Node\* tail=NULL;

void create(){

    while(1){

    Node\* new=(Node\*)malloc(sizeof(Node));

    char cha;

    printf("Enter data:");

    scanf("%d",&new->data);

    new->next=NULL;

    if(head==NULL){

        head=new;

        tail=new;

        tail->next=head;

    }

    else{

        tail->next=new;

        new->next=head;

        tail=new;

    }

    printf("Do you want to insert another node(y/n):");

    scanf(" %c", &cha);

    if(cha!='y' && cha!='Y') break;

    }

}

void display(){

    if(head==NULL){

        printf("empty LinkedList\n");

        return;

    }

    Node\* temp=head;

    printf("LinkedList content:");

    while(temp->next!=head){

        printf("%d\t",temp->data);

        temp=temp->next;

    }

    printf("%d\n",temp->data);

}

void insertatbeginning(){

    Node\* new=(Node\*)malloc(sizeof(Node));

    printf("Enter data:");

    scanf("%d",&new->data);

    if(head==NULL){

        head=new;

        tail=new;

        new->next=head;

        return;

    }

    else{

        tail->next=new;

        new->next=head;

        head=new;

        return;

    }

}

void insertatend(){

    Node\* new=(Node\*)malloc(sizeof(Node));

    printf("Enter data:");

    scanf("%d",&new->data);

    if(head==NULL){

        head=new;

        tail=new;

        new->next=head;

        return;

    }

    else{

        tail->next=new;

        new->next=head;

        tail=new;

        return;

    }

}

int length(){

    Node\* temp=head;

    int count=1;

    while(temp->next!=head){

        count++;

        temp=temp->next;

    }

    return count;

}

void insertatpos(int p){

    if(p==1){

        insertatbeginning();

        return;

    }

    int l=length();

    if(p>l+1){

        printf("Invalid position\n");

        return;

    }

    if(l==p+1){

        insertatend();

        return;

    }

    Node\* temp=head;

    int count=1;

    while(count<(p-1)){

        temp=temp->next;

        count++;

    }

    Node\* new=(Node\*)malloc(sizeof(Node));

    printf("Enter data:");

    scanf("%d",&new->data);

    new->next=temp->next;

    temp->next=new;

    return;

}

void deleteatbeginning(){

    if(head==NULL){

        printf("empty LinkedList\n");

        return;

    }

    Node\* temp=head;

    head=head->next;

    tail->next=head;

    free(temp);

    return;

}

void deleteatend(){

    if(!head){

        printf("empty LinkedList\n");

        return;

    }

    if(head->next==head){

        Node\* temp=head;

        head=NULL;

        tail=NULL;

        free(temp);

        return;

    }

    Node\* temp=head;

    while((temp->next)->next!=head){

        temp=temp->next;

    }

    Node\* last=temp->next;

    temp->next=head;

    tail=temp;

    free(last);

    return;

}

void deleteatpos(int p){

    if(!head){

        printf("empty LinkedList\n");

        return;

    }

    if(p==1){

        deleteatbeginning();

        return;

    }

    int l=length();

    if(p==l){

        deleteatend();

        return;

    }

    if(p>l){

        printf("Invalid position\n");

    }

    Node\* temp=head;

    int count=1;

    while(count<p-1){

        temp=temp->next;

        count++;

    }

    Node\* rem=temp->next;

    temp->next=temp->next->next;

    free(rem);

    return;

}

int main(){

    create();

    display();

    insertatbeginning();

    insertatend();

    insertatpos(2);

    display();

    deleteatbeginning();

    deleteatend();

    display();

    deleteatpos(2);

    display();

    return 0;

}

Output:

