1. Doubly Linked List ADT Implementation

AIM:

To implement the various ADT operations in Doubly Linked List (namely insert, delete and find operations).

PROCEDURE:

1. Start
2. Create a doubly linked list by specifying a structure with two pointers to its previous and next node, and then creating a list of nodes as such with the structure.
3. Create a switch case ladder for insert, delete and find operations:
   1. Insert Operation: Create a new node, set its data, and set its prev and next value to the appropriate nodes in the linked list.
   2. Delete Operation: Traverse to the required node, then link its previous node to the next node and null the node.
   3. Find Operation: Traverse the list for a match; on finding the node, break and return the position.
4. End

DIAGRAMMATIC REPRESENTATION:

SAMPLE INPUT/OUTPUT:

#include <stdio.h>

#include <stdlib.h>

struct node{

    int data;

    struct node \*prev;

    struct node \*next;

}\*head,\*tail;

void createList(int data){

    int i,num;

    struct node \*temp;

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

    printf("Enter the elements now: \n");

    scanf("%i",&num);

    head->data=num;

    head->prev=NULL;

    head->next=NULL;

    tail=head;

    for(int i=2;i<=data;i++){

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

        scanf("%i",&num);

        temp->data=num;

        temp->prev=tail;

        temp->next=NULL;

        tail->next=temp;

        tail=temp;

    }

}

void insertNode(int data){

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

    temp->data=data;

    temp->prev=tail;

    temp->next=NULL;

    tail->next=temp;

    tail=temp;

}

void deletenode(int data){

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

    temp=head;

    while(temp!=NULL){

        if(temp->data==data){

            //Means the deletion element is found

            temp->prev->next=temp->next;

            break;

        }else{

            temp=temp->next; }}}

void findinList(int num){

    struct node \*p;

    p=head;

    int i=0;

    while(p!=NULL){

        int data;

        data=p->data;

        if(data==num){

            printf("The required element %i found at node %i",num,i);

            break;

        }

        else{

            i++;

            p=p->next; }}}

void displayList(){

    struct node \*p;

    p=head;

    printf("\nThe list is: \n");

    while(p!=NULL){

        printf("%i ",p->data);

        p=p->next; }}

int main(){

    printf("Creating node....\n");

    printf("Enter the number of elements: ");

    int num;

    scanf("%i",&num);

    createList(num);

    printf("\n1.Add node\n2.Delete node\n3.Find node\n\n");

    int choice;

    scanf("%i",&choice);

    switch(choice){

        case 1:

        //Add node

        printf("\nEnter the number of elements to be added: ");

        int size;

        scanf("%i",&size);

        printf("\nEnter the elements now: \n");

        for(int i=0;i<size;i++){

            int temp;

            scanf("%i",&temp);

            insertNode(temp);

        }

        displayList();

        break;

        case 2:

        //Delete node

        printf("\nEnter element to be deleted: ");

        int del;

        scanf("%i",&del);

        deletenode(del);

        displayList();

        break;

        case 3:

        //Find node

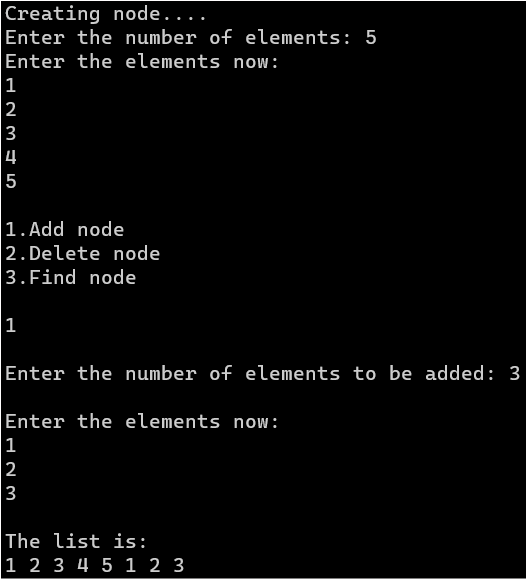
        printf("\nEnter element to be searched: ");

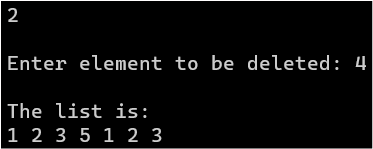
        int find;

        scanf("%i",&find);

        findinList(find);

        break; }}





RESULT:

The doubly linked list ADT has been implemented.

1. DOUBLY LINKED LIST APPLICATION – FINDING A PALINDROME

AIM:

To implement the method of finding whether a given word is a palindrome or not, using a doubly linked list.

PROCEDURE:

1. Start
2. Create a doubly linked list, with each character of the word in each node.
3. Create a pointer sequence, and set it so that left==right or left>right.
   1. Traverse the linked list as such. If only match is found, increment left by 1 and right decrement by 1.
   2. Continue in this fashion until left=SIZE and right=0.
4. Use a flag variable, to know the results.
5. End

DIAGRAMMATIC REPRESENTATION:

SAMPLE INPUT/OUTPUT:

#include <stdio.h>

#include <stdlib.h>

struct node{

    int data;

    struct node \*prev;

    struct node \*next;

}\*head,\*tail;

void createList(int data){

    int i;

    char num;

    struct node \*temp;

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

    printf("Enter the word now: \n");

    scanf("%s",&num);

    head->data=num;

    head->prev=NULL;

    head->next=NULL;

    tail=head;

    for(int i=2;i<=data;i++){

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

        scanf("%s",&num);

        temp->data=num;

        temp->prev=tail;

        temp->next=NULL;

        tail->next=temp;

        tail=temp; }}

int checkpalindrome(){

    struct node \*left,\*right;

    left=head;

    right=tail;

    while(left!=right){

        if(left->data!=right->data){

            return 0; }

        left=left->next;

        right=right->prev;}

    return 1; }

int main(){

    printf("Enter the number of elements: ");

    int num;

    scanf("%i",&num);

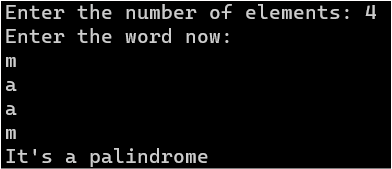
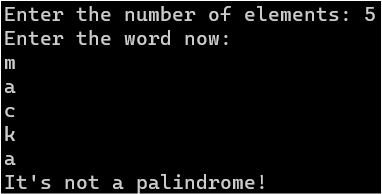
    createList(num);

    if(checkpalindrome()==1){

        printf("It's a palindrome"); }

    else{

        printf("It's not a palindrome!"); }}

RESULT:

Finding palindrome has been implemented using doubly linked list.