**Lab 5[5/2/2024]**

**1. WAP to Implement doubly link list with primitive operations  
  
a)           Create a doubly linked list.  
b)           Insert a new node to the left of the node.  
c)           Delete the node based on a specific value**

#include <stdio.h>

#include <stdlib.h>

struct node

{

struct node \*prev;

int data;

struct node \*next;

}\*head=NULL;

void display();

void create();

void insert\_left();

void delete\_node();

struct node \*new\_node,\*ptr;

void main()

{

int choice;

printf("1:create \n 2:insert\_left \n 3:delete\_node\n 4:exit");

while(1)

{

printf("\nenter choice: \t");

scanf("%d",&choice);

switch(choice)

{

case 1:

create();

break;

case 2:

insert\_left();

break;

case 3:

delete\_node();

break;

case 4:

exit(0);

default:

printf("invalid input");

}

}

}

void create()

{

int val;

printf("\nenter -1 to end");

printf("\nenter data:\t");

scanf("%d",&val);

while(val!=-1)

{

new\_node=malloc(sizeof(struct node));

new\_node->data=val;

if(head==NULL)

{

new\_node->prev=NULL;

new\_node->next=NULL;

head=new\_node;

}

else

{

ptr=head;

while(ptr->next!=NULL)

ptr=ptr->next;

ptr->next=new\_node;

new\_node->prev=ptr;

new\_node->next = NULL;

}

printf("\n enter data:\t");

scanf("%d",&val);

}

display();

}

void display()

{

if(head==NULL)

printf("the list is empty");

else

{

ptr=head;

printf("\n The list is:\n");

while(ptr!=NULL)

{

printf("%d \t",ptr->data);

ptr=ptr->next;

}

}

}

void insert\_left()

{

int val,d;

printf("enter value to be inserted:");

scanf("%d",&val);

printf("enter value before which data is to be inserted:");

scanf("%d",&d);

new\_node=malloc(sizeof(struct node));

new\_node->data=val;

if(head==NULL)

{

new\_node->prev=NULL;

new\_node->next=NULL;

head=new\_node;

}

else

{

ptr=head;

while(ptr->data!=d)

ptr=ptr->next;

if(ptr==head)

{

new\_node->prev=NULL;

new\_node->next=head;

head->prev=new\_node;

head=new\_node;

}

else

{

new\_node->next=ptr;

new\_node->prev=ptr->prev;

ptr->prev->next=new\_node;

ptr->prev=new\_node;

}

}

display();

}

void delete\_node()

{

if(head==NULL)

printf("\n the list is empty");

else

{

int val;

printf("enter value to be deleted:\n");

scanf("%d",&val);

ptr=head;

while(ptr->data!=val)

ptr=ptr->next;

if(ptr==head)

{

head=head->next;

head->prev=NULL;

free(ptr);

}

else

{

ptr->prev->next=ptr->next;

ptr->next->prev=ptr->prev;

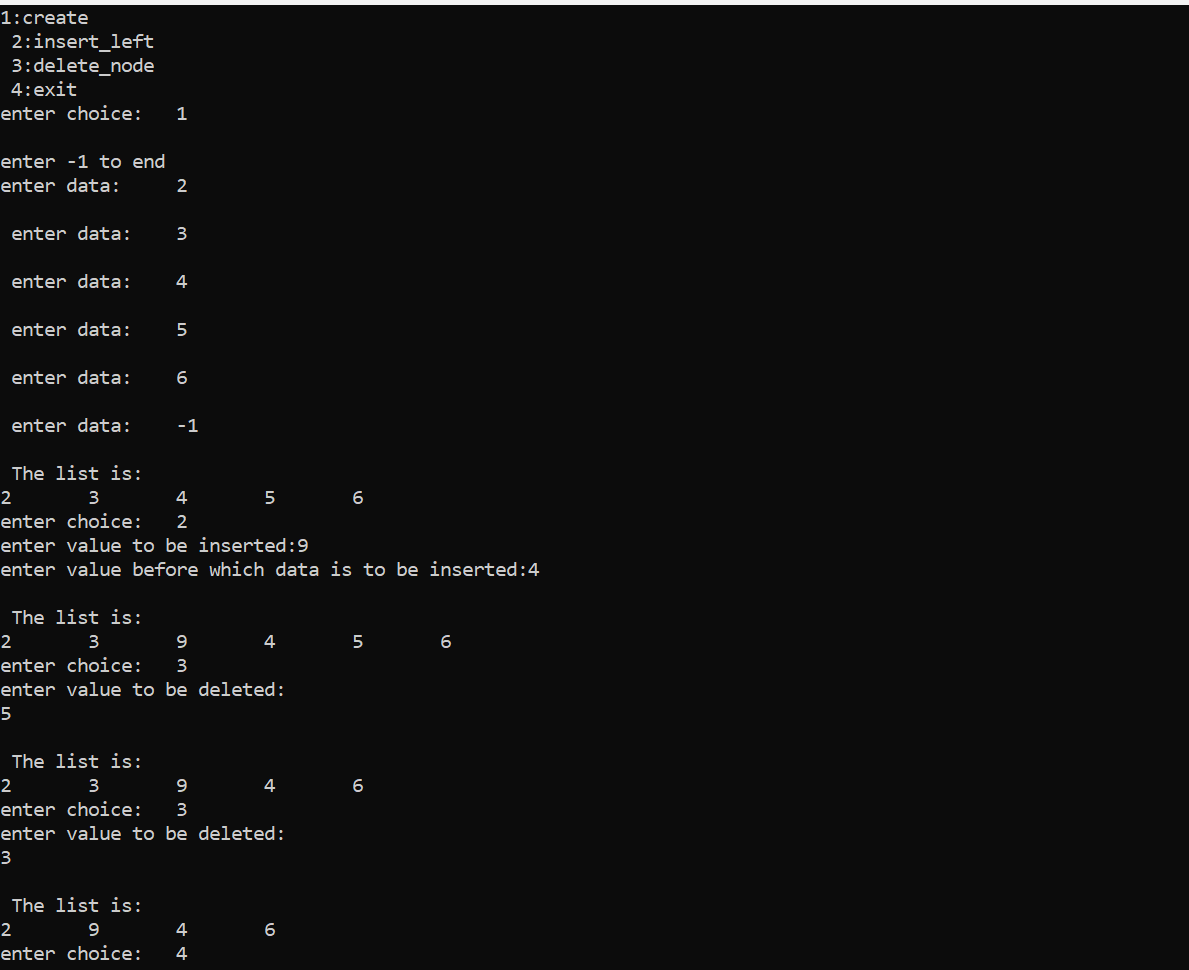
free(ptr);

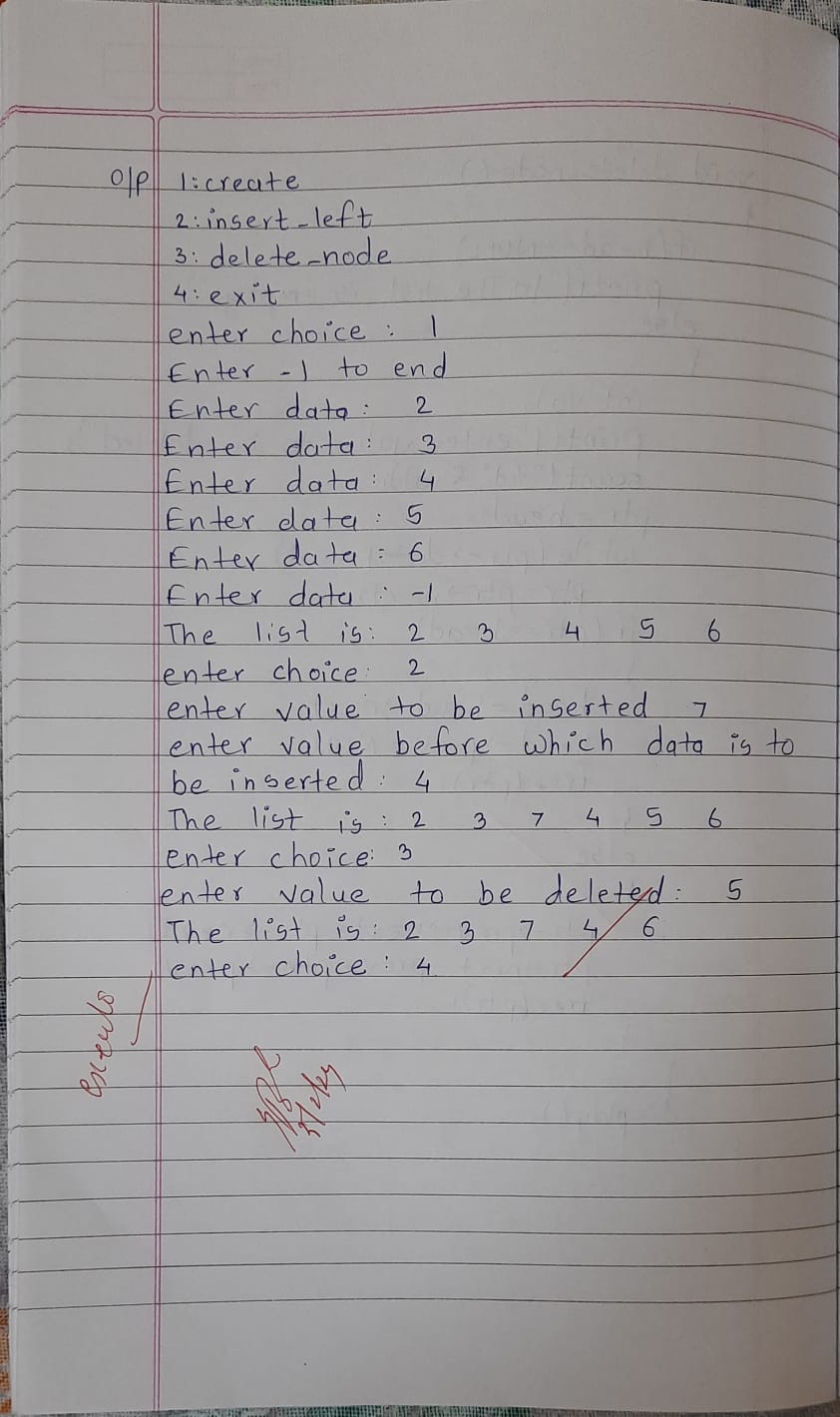
}

}

display();

}





LEETCODE PROGRAM:

int scoreOfParentheses(char\* s) {

    if (s == NULL || \*s == '\0')

        return 0;

    int stack[50];

    int top = 0;

    int score = 0;

    for (int i = 0; s[i] != '\0'; i++) {

        if (s[i] == '(') {

            if (top >= 50) {

                return 0;

            }

            stack[top++] = score;

            score = 0;

        } else {

            int previousScore = stack[--top];

            if (score > 0) {

                score = previousScore + 2 \* score;

            } else {

                score = previousScore + 1;

            }

        }

    }

    if (top != 0) {

        return 0;

    }

    return score;

}

