5/2/24

Lab - 6

1) Write a program to implement doubly linked list with primitive operations
a) create doubly linked list
b) insert a new node to the left of node
c) delete the node based on a specific value

ans)

```
# include <stdio.h>
# include <stdlib.h>
struct node
{
    int data;
    struct node * prev;
    struct node * next;
};
struct node * head = NULL;

void createlist ()
{
    int i, n;
    struct node * newnode;
    struct node * temp;
    printf (" Enter the no. of elements: ");
    scanf (" %d", &n);
    for (i = 0; i < n; i++)
    {
        newnode = (struct node *) malloc (sizeof
                                (struct node));
        printf (" Enter the element : ");
        scanf (" %d", & newnode -> data);
        if (head == NULL)
        {
```

```
        head = temp = newnode;
        head -> prev = NULL;
        head -> next = NULL;
    }
    else
    {
        temp -> next = newnode;
        newnode -> prev = temp;
        temp = newnode;
        temp -> next = NULL;
    }
}
printf("List created successfully! \n");
}
void insert (int target, int data)
{
    struct node * temp = head;
    while (temp != NULL)
    {
        if (temp -> data == target)
        {
            struct node * newnode = (struct node*)
                malloc (sizeof (struct node));
            newnode -> data = data;
            newnode -> next = temp;
            newnode -> prev = temp -> prev;
            if (temp -> prev != NULL)
            {
                temp -> prev -> next = newnode;
            }
            temp -> prev = newnode;
            if (head == temp)
            {
                head = newnode;
            }
    }
```

```
        temp = temp -> next;
    }
    if (temp == NULL)
    {
        printf("Target node doesn't exist");
        return;
    }
    printf("Node inserted");
}
void delete(int value)
{
    struct node * current = head;
    while (current != NULL)
    {
        if (current -> data == value)
        {
            if (current -> prev != NULL)
            {
                current -> prev -> next = current ->
            }
            if (current -> next != NULL)
            {
                current -> next -> prev = current ->
            }
            if (current == head)
            {
                head = current -> next;
            }
            free (current);
            printf("Node deleted");
            return;
        }
        current = current -> next;
    }
```

```
        printf("Node not found");
}

void display()
{
    struct node * temp = head;
    if (temp == NULL)
    {
        printf("List is empty");
        return;
    }
    printf("The list is : ");
    while (temp != NULL)
    {
        printf("%d", temp -> data);
        temp = temp -> next;
    }
    printf("\n");
}

void main()
{
    int choice, data, target, value;
    printf("\nMenu\n1. Create linked list\n
    2. Insert node\n3. Delete node\n4.
    Display list\n5. Exit\n");
    while(1)
    {
        printf("Enter your choice :");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1:
                createlist();
                break;
```

```
case 2:
    printf("Enter the value of the
    node to insert left of:");
    scanf("%d", &target);
    printf("Enter the element to
    insert:");
    scanf("%d", &data);
    insert(target, data);
    break;
case 3:
    printf("Enter the value of the
    node to delete:");
    scanf("%d", &value);
    delete(value);
    break;
case 4:
    display();
    break;
case 5:
    exit(0);
    break;
default:
    printf("Invalid choice");
  }
 }
}
```

OUTPUT:

Menu

1. Create linked list
2. Insert node
3. Delete node
4. Display
5. Exit

Enter your choice : 1
Enter the no. of elements : 3
Enter the element : 2
Enter the element : 4
Enter the element : 6
List created successfully
Enter your choice : 3
Enter the value of the node to delete : 4
Enter your choice : 4
The list is : 2 6
Enter your choice : 5

19/2/24

2) Given a balanced parenthesis string s, return the score of that string

ans)

```
#include<stdio.h>
#include<stdlib.h>
int scoreofparanthesis(char *s)
{
    int stack[50];
    int top=-1, score=0;
    for(int i=0; s[i]!='\0'; i++)
    {
        if(s[i]=='(')
        {
            stack[++top]=score;
            score=0;
        }
        else
        {
```

19/2/24

```
MENU:
1. Create linked list
2. Insert left of node
3. Delete node by value
4. Print the list
5. Exit
Enter your choice: 1
Enter the number of elements:3
Enter the element: 2
Enter the element: 4
Enter the element: 6
List created successfully.

MENU:
1. Create linked list
2. Insert left of node
3. Delete node by value
4. Print the list
5. Exit
Enter your choice: 2
Enter the value of the node to insert left of: 4
Enter the element to insert left of the node: 3
Node inserted successfully.
```

```
MENU:
1. Create linked list
2. Insert left of node
3. Delete node by value
4. Print the list
5. Exit
Enter your choice: 3
Enter the value of the node to delete: 6
Node deleted successfully.

MENU:
1. Create linked list
2. Insert left of node
3. Delete node by value
4. Print the list
5. Exit
Enter your choice: 4
Doubly linked list: 2 3 4

MENU:
1. Create linked list
2. Insert left of node
3. Delete node by value
4. Print the list
5. Exit
Enter your choice: 5
```