Week-6

URBAN EDGE 21/02/24

// 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 base on a specific value

```
#include <stdio.h>
#include <stdlib.h>

Struct Node
{
    int data;
    Struct node *prev;
    Struct Node *next;
};

Struct Node *createNode (int data)
{
    Struct node *newNode = (Struct Node*) malloc (Sizeof (StructNode));
    if (newnode == NULL)
    {
        printf ("memory allocation failed \n");
        exit (1);
    }
    newNode -> data = data;
    newNode -> prev = NULL;
    newNode -> next = NULL;
    Return newNode;
};
```

```
Void insertNode (Struct node *head, Struct node *forget,
                 int data)
{
    Struct Node *newnode = createNode (data);
    if (forget -> prev != NULL)
    {
        forget -> prev -> next = newnode;
        newnode -> prev = forget -> prev;
    }
    else
    {
        head = newnode;
    }
    newnode -> next = forget;
    forget -> prev = newnode;
}
Void deleteNode (Struct node *head, int value)
{
    Struct Node *current = head;
    while (current != NULL)
    {
        if (current -> data == NULL)
        {
            if (current -> prev -> next = current -> next;
        }
        else
        {
            head = current -> next;
        }
        if (current -> prev != NULL)
        {
            current -> next -> prev = current -> prev;
        }
        free (current);
        return;
    }
```



```
    current = current -> next;
    }
    printf (" node with value %d not found \n");
}

Void display (Struct node *head)
{
    printf (" doubly linked list : \n");
    while (head != NULL)
    {
        printf ("%d <-> ", head -> data);
        head = head -> next;
    }
    printf (" NULL \n");
}

Void main()
{
    Struct node *head = NULL;
    head = createNode (1);
    head -> next = createNode (2);
    head -> next -> prev = head;
    head -> next -> next = createNode (3);
    head -> next -> next -> prev = head -> next;

    display (head);
    insertNode (head, head -> next, 10);
    printf (" after insertion : \n");
    display (head);

    deleteNode (head, 2);
    printf (" after deletion : \n");
    display (head);
    return 0;
}
```

Output:

doubly linked list

1 <-> 2 <-> 3 <-> NULL

after insertion:

doubly linked list

1 <-> 10 <-> 2 <-> 3 <-> NULL

after deletion:

doubly linked list

1 <-> 10 <-> 3 <-> NULL