5224

LAB-6

Date
Page

(07) 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 {
    int data;
    struct node* next;
    struct node* prev;
};
struct node *head=0, *newnode, *temp;
void create()
{   int i, n;
    printf ("Enter the no. of elements : \n");
    scanf ("%d", &n);
    for (i=0; i<n; i++)
    {   newnode = (struct node*) malloc( sizeof (struct node));
        printf ("Enter the %d element : \n", i+1);
        scanf ("%d", &newnode->data );
        newnode -> prev = 0;
        newnode -> next = 0;
        if (head == 0)
        { temp = head = newnode; }
        else {
            temp -> next = newnode;
            newnode -> prev = temp;
            temp = newnode; }
    }
}
```

O/P

enter operation

(1) create (2) display (3) insert at left. (4) delete at rear

-> 1

Enter no. of elements

3

Enter 1 element: 1

Enter 2 element: 2

Enter 3 element: 3

Enter operation: 3

enter pole: 2

Enter data 10

Enter operation 2

1

10

2

3

Enter operation

-1

completed

```
void delete_pos() {
    int pos, i=1;
    temp = head;
    printf("Enter position \n");
    scanf("%d", &pos);
    while (i < pos) {
        temp = temp -> next;
        i++;
    }
    temp -> prev -> next = temp -> next;
    temp -> next -> prev = temp -> prev;
    free(temp);
}
void main() {
    int choice, num;
    printf("Enter operation: \n 1. create \n 2. display \n
        3. insert at left \n 4. delete at position \n
        \n 5. -1 to exit \n");
while (1) {
    printf("Enter operation \n");
    scanf("%d", &choice);
    if (choice == 1) {
        printf("completed \n"); }
else {
    switch (choice) {
        case 1: create();
        break;
        case 2: display();
        break;
        case 3: insert_left();
        break;
        case 4: delete_pos();
        break;
default: printf("Invalid input \n"); }}}
```

```
void display() {
    temp = head;
    while (temp != 0) {
        printf ("%d \n", temp->data);
        temp = temp -> next;
    }
}
void insert_left() {
    int node, i = 1;
    printf ("enter the node \n");
    scanf ("%d", &node);
    temp = head;
    if (node < 1) {
        printf ("invalid position \n"); }
    else if (node == 1) {
    newnode = (struct node *) malloc (sizeof(struct node));
    printf ("enter data \n");
    scanf ("%d", &newnode->data);
    newnode -> prev = 0;
    head -> prev = newnode;
    newnode -> next = head;
    head = newnode; }
    else {
        newnode = (struct node *) malloc (sizeof(struct node));
        printf ("enter data\n");
        scanf ("%d", &newnode->data;
        while (i < node-1)
        { temp = temp -> next;
            i++; }
        newnode -> prev = temp;
        newnode -> next = temp -> next;
        temp -> next = newnode;
        newnode -> next -> prev = newnode; } }
```