1.

Expt-1

Linked List

Aim

write a menu driven program for performing the following operations on a linked list. 1. Display 2. Insert at beginning. 4. Insert at specified position. 5. Delete from beginning 6. Delete from End 7. Delete from specified position.

Algorithms

Step 1: Start

Step 2: Create a structure node with data and node. pointer link.

Step 3: Create a node. pointer head and initalize. to NULL.

Step 4: Declare alen() function.

i) intialize temp = head; length = 0

ii) repeat steps (iii)(iv) below till temp != null.

iii) length ++;

iv = temp = temp -> link.

v return length,

Step 5: Declare display

i) temp = head

ii) repeat iv till temp != NULL.

iii) print temp -> data.

iv temp = temp -> link.

step 6: Delare add_end fn with data as input.

i) Intialize temp->hed.

ii) make a newnode, newnod->data=dat

iii) newnode->link=NULL

iii) if (tem!=NULL). repeat step blat. till temp->link != NULL

iv tem = temp->link,

temp->link= newnode

else.

vii else head=newnode.

step 7: Declare add_beg (int data).

i) Create newnode, newnode->data=data.

ii) newnod->link= head.

iii) head =newnode

step 8: Declare add_pos (int data, int pos).

i) node * temp=head

ii create newnode newnod->data = data.

iii if (pos<=0 && pos > lon'hod

point invalid.

iv. else if (po->1)

repeat step below till --pos > 1 && temp->link != NULL).

vi) temp = temp->link.

vii newnode->link = temp->link.

temp->link = newnode.

ix else add_beg (data).

step 9: Declare del_beg().

i) if(head != NULL) head = head -> link;

ii) else. printf("LL is empty

step 10: Declare del_end() to delete at end.

i) temp = head.

ii) if(temp == NULL) print empty

iii) else.

iv) if(head -> link != NULL).

v) repeat #step below tom temp -> link != NULL.

vi) temp = temp -> link.

vii) temp -> = NULL

else head = NULL

step 11. Declare del pos (int pos).

i) node * temp = head;

ii) if(pos <= 0 || pos > len(head) print invalid.

iii) else if(pos > 1).

repeat step below till --pos > 1 &&
temp != NULL.

temp = temp -> link.

iv) temp -> link = temp -> link -> link

v) else del beg().

Step 12 Declare main function..

Step 13 ~~Declare~~ a menu driven program to call the necessary function

Step 14: Stop.

Code

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

struct node
{
    int data;
    struct node *link;
};

struct node *head = NULL;

int length()
{
    struct node *temp = head;
    int length = 0;
    while (temp != NULL)
    {
        temp = temp->link;
        length++;
    }
    return length;
}

void display()
{
    struct node *temp = head;
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->link;
    }
    printf("\n");
}

void add_end(int data)
{
    struct node *temp = head;
    struct node *newnode = (struct node *)malloc(sizeof(struct node));

    newnode->data = data;

    newnode->link = NULL;

    if (temp != NULL)
    {
        while (temp->link != NULL)
        {
```

```
            temp = temp->link;
        }
        temp->link = newnode;
    }
    else
    {

        head = newnode;
    }
}

void add_beg(int data)
{
    struct node *newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = data;
    newnode->link = head;
    head = newnode;
}

void add_pos(int data, int pos)
{
    struct node *temp = head;
    struct node *newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = data;

    if (pos > length() || pos < 1)
    {
        printf("Invalid Position\n");
    }
    if (pos > 1)
    {
        while (--pos > 1 && temp != NULL)
        {
            temp = temp->link;
        }

        newnode->link = temp->link;
        temp->link = newnode;
    }
    else
    {
        add_beg(data);
    }
}

void del_beg()
{
    if (head != NULL)
```

```
    {
        head = head->link;
    }
    else
    {
        printf("Linkd List is empty\n");
    }
}

void del_end()
{
    struct node *temp = head;
    if (temp == NULL)
    {
        printf("The Linked List is empty\n");
    }
    else
    {
        if (head->link != NULL)
        {
            while (temp->link->link != NULL)
            {
                temp = temp->link;
            }
            temp->link = NULL;
        }
        else
        {
            head = NULL;
        }
    }
}

void del_pos(int pos)
{
    struct node *temp = head;
    // if(pos>length() || pos<1)
    // {
    //  printf("Invalid Position\n");
    // }
    if (pos > 1)
    {

        while (--pos > 1 && temp != NULL)
        {
            temp = temp->link;
        }
```

```
        temp->link = temp->link->link;
    }
    else
    {
        del_beg();
    }
}
int main()
{

    while (1)
    {

        int choice;
        printf("1.Display\n2.Insert at End\n3.Insert at Beginning\n4.Delete
from beginning\n5.Delete from end\n6.Insert into position\n7.Delete from
position\n8.Exit");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
        {
            display();
            break;
        }
        case 2:
        {
            int temp;
            printf("Enter the data");
            scanf("%d", &temp);
            add_end(temp);

            display();

            break;
        }
        case 3:
        {
            int temp;
            printf("Enter the data");
            scanf("%d", &temp);
            add_beg(temp);
            display();
            break;
        }
        case 6:
        {
            int temp, pos;
```

```
            printf("Enter the data and position");
            scanf("%d %d", &temp, &pos);
            add_pos(temp, pos);
            display();
            break;
        }
        case 4:
        {
            del_beg();
            display();
            break;
        }
        case 5:
        {
            del_end();
            display();
            break;
        }
        case 7:
        {
            int pos;
            printf("Enter the position");
            scanf("%d", &pos);
            del_pos(pos);
            display();
            break;
        }

        case 8:
        {
            return 0;
        }
        }
    }
    return 0;
}
```

OUTPUT

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit2

Enter the data12

12

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit2

Enter the data13

12 13

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit3

Enter the data11

11 12 13

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit6

Enter the data and position10 2

11 10 12 13

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit4

10 12 13

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit5

10 12

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit7

Enter the position2

10

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit8