**EXERCISE 3**

# LINKED LIST

1. **AIM:**

To create a singly linked list and to insert, delete the element from the first, last and middle position.

**CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node\* create(int num);

void addlist(struct node\* s);

void traverse();

void insert\_first();

void insert\_last();

void insert\_middle();

void delete\_first();

void delete\_last();

void delete\_middle();

void delete\_value();

struct node

{

int data;

struct node \*next;

};

struct node \*temp,\*head=NULL;

int count;

int main()

{

int num,n,i;

struct node \*s1;

printf("enter n:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("enter the num:");

scanf("%d",&num);

s1=create(num);

addlist(s1);

}

printf("the list is:");

traverse();

insert\_first();

insert\_last();

insert\_middle();

delete\_first();

delete\_last();

delete\_middle();

delete\_value();

}

struct node\* create(int num)

{

struct node \*s;

s=(struct node\*)malloc(sizeof(struct node));

s->data=num;

s->next=NULL;

return s;

}

void addlist(struct node \*s)

{

if(head==NULL)

{

head=s;

temp=s;

}

else

{

temp->next=s;

temp=temp->next;

}

}

void traverse()

{

count=0;

temp=head;

while(temp!=NULL)

{

printf("%d",temp->data);

temp=temp->next;

count++;

}

}

void insert\_first()

{

struct node \*s;

s=(struct node\*)malloc(sizeof(struct node));

printf("\n enter the number to be inserted at first:");

scanf("%d",&s->data);

if(head==NULL)

{

head=s;

s->next=NULL;

}

else

{

s->next=head;

head=s;

}

printf("\n the list after insertion at first is:");

traverse();

}

void insert\_last()

{

struct node \*s;

s=(struct node\*)malloc(sizeof(struct node));

printf("\n enter the number to be inserted at last:");

scanf("%d",&s->data);

if(head==NULL)

{

head=s;

s->next=NULL;

}

else

{

temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=s;

s->next=NULL;

printf("\n the list after insertion at last is:");

traverse();

}}

void insert\_middle()

{

int mid,i=1;

struct node \*temp1;

if(head==NULL)

{

printf("the list is empty");

}

else

{

mid=count/2;

temp=head;

struct node \*s;

s=(struct node\*)malloc(sizeof(struct node));

printf("\n enter the number to be inserted at middle:");

scanf("%d",&s->data);

while(i<mid)

{

temp=temp->next;

i++;

}

s->next=temp->next;

temp->next=s;

printf("\n the list after insertion at middle:");

traverse();

}}

void delete\_first()

{

if(head==NULL)

{

printf("the list is empty");

}

else

{

temp=head;

head=head->next;

free(temp);

printf("\n list after deletion at first:");

traverse();

}}

void delete\_last()

{

if(head==NULL)

{

printf("the list is empty");

}

else

{

temp=head;

while(temp->next->next!=NULL)

{

temp=temp->next;

}

temp->next=NULL;

printf("\n list after deletion at last is");

traverse();

}}

void delete\_middle()

{

int mid,i=0;

struct node \*temp1;

temp=head;

mid=count/2;

temp1=head;

while(i<mid-1)

{

temp=temp->next;

i++;

}

i=0;

while(i<mid)

{

temp1=temp1->next;

i++;

}

temp->next=temp1->next;

printf("\n list after deleting the middle element:");

traverse();

}

void delete\_value()

{

if(head==NULL)

{

printf("the list is empty");

}

else

{

int num,i=0;

struct node \*temp1;

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

scanf("%d",&num);

temp=head;

temp1=head;

while(temp!=NULL)

{

if(head->data==num)

{

delete\_first();

break;

}

else if(temp->data==num)

{

temp1->next=temp->next;

break;

}

temp=temp->next;

if(i>=1)

{

temp1=temp1->next;

}

i++;

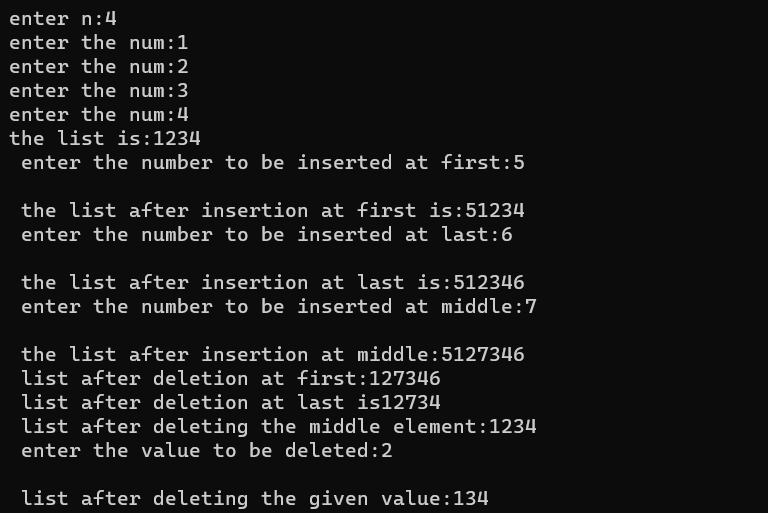
}

printf("\n list after deleting the given value:");

traverse();

}}

**OUTPUT:**

****

**2)AIM:**

To create a doubly linked list and to insert, delete the element from the first, last and middle position.

**CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

struct node \*pre;

};

typedef struct node stn;

stn \*head=NULL,\*last=NULL;

int count;

stn\* create(int num);

void addList(stn \*s1);

void traverse();

void reverse();

void insert\_first();

void insert\_last();

void insert\_middle();

void delete\_first();

void delete\_last();

void delete\_middle();

void delete\_element();

int main()

{

stn \*s1;

int i,num,n;

printf("enter n:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("enter the element:");

scanf("%d",&num);

s1=create(num);

addList(s1);

}

traverse();

reverse();

insert\_first();

insert\_last();

insert\_middle();

delete\_first();

delete\_last();

delete\_middle();

delete\_element();

}

stn\* create(int num1)

{

stn \*s;

s=(stn\*)malloc(sizeof(stn));

s->data=num1;

s->next=NULL;

s->pre=NULL;

}

void addList(stn\* s1)

{

if(head==NULL)

{

head=s1;

last=head;

}

else

{

s1->pre=last;

last->next=s1;

last=last->next;

}

last=s1;

last->next=NULL;

}

void traverse()

{

stn \*temp;

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

temp=head;

count=0;

while(temp!=NULL)

{

printf("%d",temp->data);

temp=temp->next;

count++;

}

}

void reverse()

{

stn \*temp;

printf("\n the reverse order of list is:");

temp=last;

while(temp!=NULL)

{

printf("%d",temp->data);

temp=temp->pre;

}

}

void insert\_first()

{

stn\* s;

s=(stn\*)malloc(sizeof(stn));

printf("\n enter the num insert at first:");

scanf("%d",&s->data);

s->next=head;

head->pre=s;

s->pre=NULL;

head=s;

printf("\n the list after insertion at first is:");

traverse();

}

void insert\_last()

{

stn\* s;

s=(stn\*)malloc(sizeof(stn));

printf("\n enter the num to insert at last:");

scanf("%d",&s->data);

last->next=s;

s->pre=last;

s->next=NULL;

last=s;

printf("\n the list after insertion at last is:");

traverse();

}

void insert\_middle()

{

stn \*s,\*temp,\*temps;

int i,mid;

s=(stn\*)malloc(sizeof(stn));

printf("\n enter the num to insert at middle:");

scanf("%d",&s->data);

mid=count/2;

temp=head;

for(i=0;i<mid;i++)

{

temp=temp->next;

}

s->pre=temp->pre;

temps=temp->pre;

s->next=temp;

temp->pre=s;

temps->next=s;

printf("\n the list after insertion at middle is:");

traverse();

}

void delete\_first()

{

stn \*temp;

temp=head;

head=head->next;

head->pre=NULL;

free(temp);

printf("\n the list after deletion at first is:");

traverse();

}

void delete\_last()

{

stn \*temp;

temp=last;

last=last->pre;

last->next=NULL;

free(temp);

printf("\n the list after deletion at the last is:");

traverse();

}

void delete\_middle()

{

int mid,i;stn \*temp;

mid=count/2;

temp=head;

for(i=0;i<mid;i++)

{

temp=temp->next;

}

temp->pre->next=temp->next;

temp->next->pre=temp->pre;

free(temp);

printf("\n the list after deletion at the middle is:");

traverse();

}

void delete\_element()

{

int num;

stn \*temp;

printf("\n enter the element to delete:");

scanf("%d",&num);

temp=head;

while(temp!=0)

{

if(head->data==num)

{

delete\_first();

break;

}

else if(temp->data==num)

{

temp->pre->next=temp->next;

temp->next->pre=temp->pre;

break;

}

else if(last->data==num)

{

delete\_last();

break;

}

else

{

temp=temp->next;

}

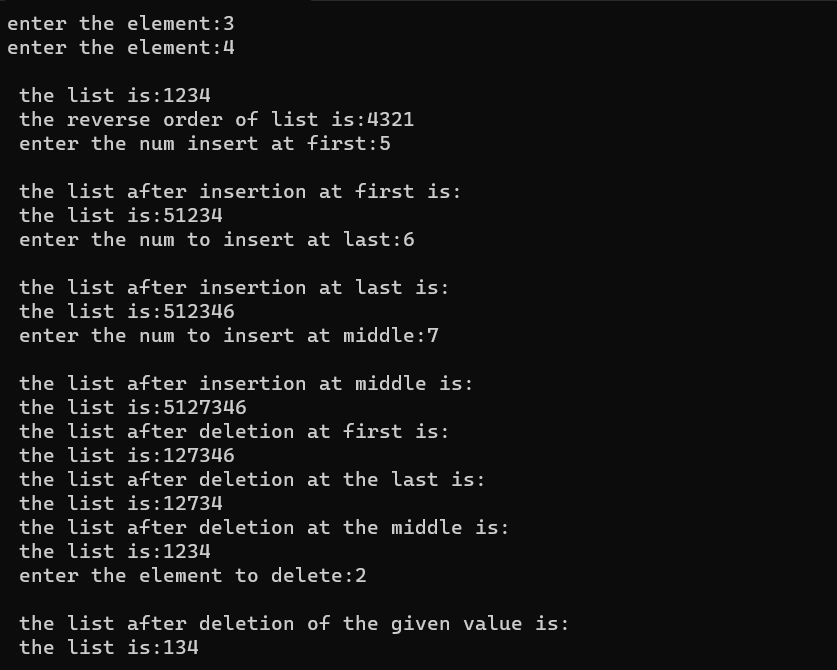
}

printf("\n the list after deletion of the given value is:");

traverse();

}

**OUTPUT:**

****