WEEK-7

Date:06-08-2025

List of programs:

1. Write a C program to insert a node at the beginning in a Double linked list.
2. Write a C program to insert a node at the end in a Double linked list.
3. Write a C program to insert a node after a given node(middle case) in a Double linked list.
4. Write a C program to delete a node at the beginning in a Double linked list.
5. Write a C program to delete a node at the end in a Double linked list
6. Write a C program to delete a node after a given node(middle case) in a Double linked list.

1. **Aim:** To write a C program to insert a node at the beginning in a Double linked list.

Program:

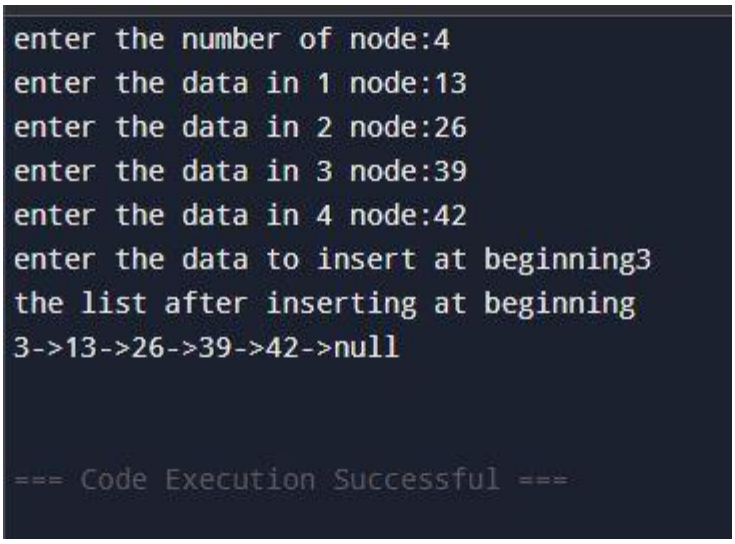
```
#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node *next;

struct node *prev;

};

struct node *head,*temp,*new,*last=NULL;

struct node *getnode(int x)

{

struct node *temp=(struct node*)malloc(sizeof(struct node));

temp->data=x;

temp->next=NULL;
```

```
temp->prev=NULL;
return temp;
}
void insert_begin(int x)
{
new=getnode(x);
if(head==NULL)
   head=last=new;
else
{
new->next=head;
head->prev=new;
head=new;
}
}
void display()
{
if(head==NULL)
{
printf("list empty\n");
}
else
{
temp=head;
while(temp!=NULL)
{
printf("%d->",temp->data);
temp=temp->next;
```

```
}
}
printf("null\n");
}
int main()
{
int n,i,x;
printf("enter the number of node:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter the data in %d node:",i+1);
scanf("%d",&x);
new=getnode(x);
if(head==NULL)
{
head=new;
}
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=new;
new->prev=temp;
last=new;
```

```
}

}

printf("enter the data to insert at beginning");

scanf("%d",&x);

printf("the list after inserting at beginning\n");

insert_begin(x);

display();

return 0;

}
```

Output:



```
enter the number of node:4
enter the data in 1 node:13
enter the data in 2 node:26
enter the data in 3 node:39
enter the data in 4 node:42
enter the data to insert at beginning3
the list after inserting at beginning
3->13->26->39->42->null



=== Code Execution Successful ===
```

2. **Aim:** Write a C program to insert a node at the end in a Double linked list.

Program:

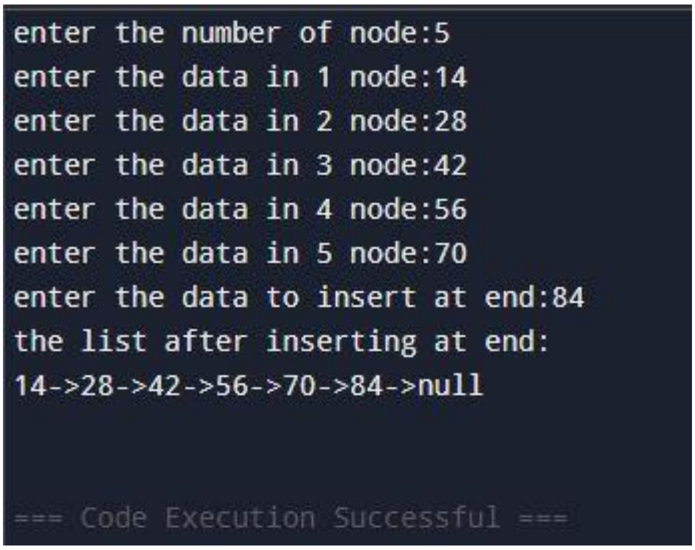
```
#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node *next;

struct node *prev;

};

struct node *head,*temp,*new,*last;

struct node *getnode(int x)

{

struct node *temp=(struct node*)malloc(sizeof(struct node));

temp->data=x;

temp->next=NULL;

temp->prev=NULL;

return temp;

}

void insert_end(int x)

{

new=getnode(x);

if(head==NULL)

   head=last=new;

else
```

```
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
last=temp;
last->next=new;
new->prev=last;
last=new;
}
}
void display()
{
if(head==NULL)
{
printf("list empty\n");
}
else
{
temp=head;
while(temp!=NULL)
{
printf("%d->",temp->data);
temp=temp->next;
}
}
printf("null\n");
```

```
}
int main()
{
int n,i,x;
printf("enter the number of node:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter the data in %d node:",i+1);
scanf("%d",&x);
new=getnode(x);
if(head==NULL)
{
head=last=new;
}
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=new;
new->prev=temp;
last=new;
last->next=NULL;
}
}
```

```
printf("enter the data to insert at end:");
scanf("%d",&x);
printf("the list after inserting at end:\n");
insert_end(x);
display();
return 0;
}
```

Output:



```
enter the number of node:5
enter the data in 1 node:14
enter the data in 2 node:28
enter the data in 3 node:42
enter the data in 4 node:56
enter the data in 5 node:70
enter the data to insert at end:84
the list after inserting at end:
14->28->42->56->70->84->null


=== Code Execution Successful ===
```

3. **Aim:** Write a C program to insert a node after a given node(middle case) in a Double linked list.

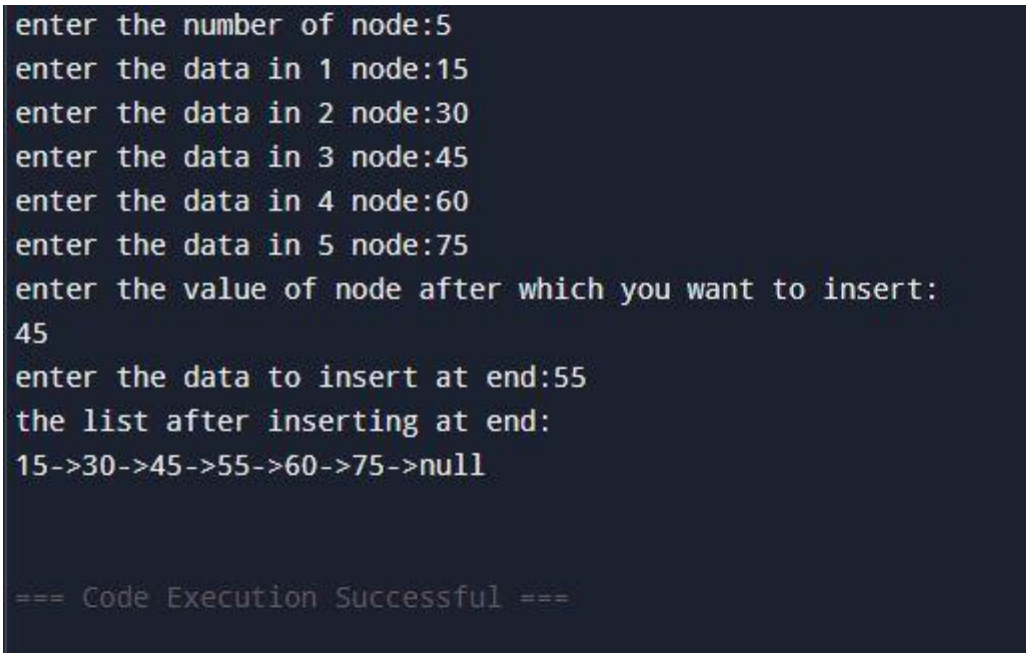
Program:

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node *next;

struct node *prev;

};

struct node *head,*temp,*new,*last=NULL;

struct node *getnode(int x)

{

struct node *temp=(struct node*)malloc(sizeof(struct node));

temp->data=x;

temp->next=NULL;

temp->prev=NULL;

return temp;

}

void insert_middle(int x,int val)

{

new=getnode(x);

if(head==NULL)

   head=last=new;
```

```
else if(last->data==val)
{
last->next=new;
new->prev=last;
last=new;
}
else
{
temp=head;
while(temp->data!=val&&temp!=NULL)
{
temp=temp->next;
}
new->next=temp->next;
(temp->next)->prev=new;
temp->next=new;
new->prev=temp;
}
}
void display()
{
if(head==NULL)
{
printf("list empty\n");
}
else
{
temp=head;
```

```
while(temp!=NULL)
{
printf("%d->",temp->data);
temp=temp->next;
}
}
printf("null\n");
}
int main()
{
int n,i,x,val;
printf("enter the number of node:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter the data in %d node:",i+1);
scanf("%d",&x);
new=getnode(x);
if(head==NULL)
{
head=last=new;
}
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=new;
new->prev=temp;
last=new;
last->next=NULL;
}
}
printf("enter the value of node after which you want to insert:\n");
scanf("%d",&val);
printf("enter the data to insert at end:");
scanf("%d",&x);
printf("the list after inserting at end:\n");
insert_middle(x,val);
display();
return 0;
}
```

Output:



```
enter the number of node:5
enter the data in 1 node:15
enter the data in 2 node:30
enter the data in 3 node:45
enter the data in 4 node:60
enter the data in 5 node:75
enter the value of node after which you want to insert:
45
enter the data to insert at end:55
the list after inserting at end:
15->30->45->55->60->75->null



=== Code Execution Successful ===
```

4. **Aim:** To write a C program to delete a node at the beginning in a Double linked list.

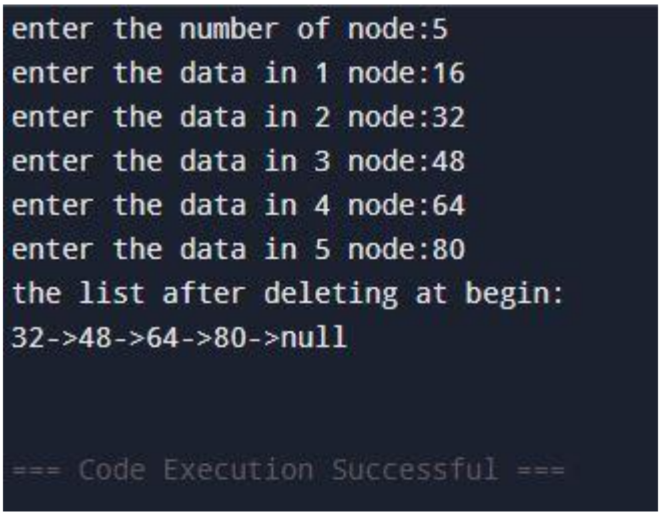
Program:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
struct node *prev;
};
struct node *head,*temp,*new,*last;
struct node *getnode(int x)
{
struct node *temp=(struct node*)malloc(sizeof(struct node));
temp->data=x;
temp->next=NULL;
temp->prev=NULL;
return temp;
}
void delete_begin()
{
if(head==NULL)
 {
   printf("list is empty");
 }
else if(head->next==NULL)
{
 temp=head;
head=last=NULL;
free(temp);
}
else
{
temp=head;
head=head->next;
head->prev=NULL;
free(temp);
}
}
void display()
{
if(head==NULL)
{
printf("list empty\n");
}
else
{
temp=head;
while(temp!=NULL)
{
printf("%d->",temp->data);
temp=temp->next;
```

```
}
}
printf("null\n");
}
int main()
{
int n,i,x;
printf("enter the number of node:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter the data in %d node:",i+1);
scanf("%d",&x);
new=getnode(x);
if(head==NULL)
{
head=last=new;
}
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=new;
new->prev=temp;
last=new;
```

```
last->next=NULL;
}
}
printf("the list after deleting at begin:\n");
delete_begin();
display();
return 0;
}
```

Output:



```
enter the number of node:5
enter the data in 1 node:16
enter the data in 2 node:32
enter the data in 3 node:48
enter the data in 4 node:64
enter the data in 5 node:80
the list after deleting at begin:
32->48->64->80->null


=== Code Execution Successful ===
```

5. **Aim:** Write a C program to delete a node at the end in a Double linked list.

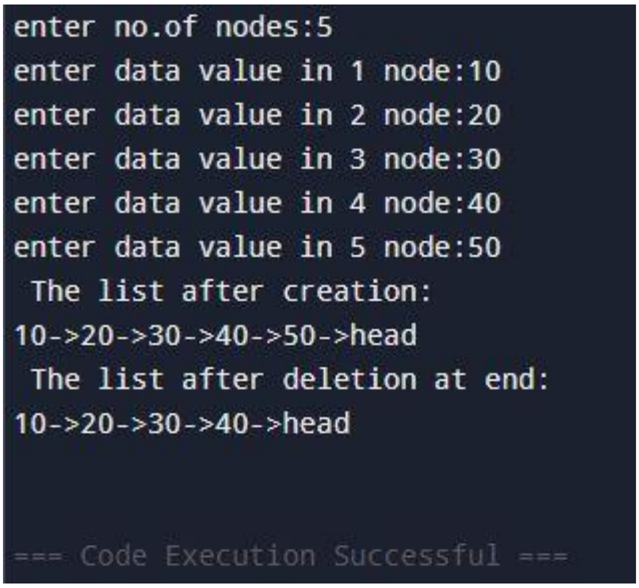
Program:

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node *next;

struct node *prev;

};

struct node *head,*temp,*new,*last=NULL;

struct node *getnode(int x)

{

struct node *temp=(struct node*)malloc(sizeof(struct node));

temp->data=x;

temp->next=NULL;

temp->prev=NULL;

}

void delete_end()

{

if(head==NULL)

 {

   printf("list is empty");

 }

else if(head->next==NULL)
```

```
{
 temp=head;
head=last=NULL;
free(temp);
}
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp=last;
last=last->prev;
last->next=NULL;
free(temp);
}
}
void display()
{
if(head==NULL)
{
printf("list empty\n");
}
else
{
temp=head;
while(temp!=NULL)
```

```
{
printf("%d->",temp->data);
temp=temp->next;
}
}
printf("null\n");
}
int main()
{
int n,i,x;
printf("enter the number of node:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter the data in %d node:",i+1);
scanf("%d",&x);
new=getnode(x);
if(head==NULL)
{
head=last=new;
}
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=new;
new->prev=temp;
last->next=NULL;
}
}
printf("the list after deleting at end:\n");
delete_end();
display();
return 0;
}
```

Output:



```
enter no.of nodes:5
enter data value in 1 node:10
enter data value in 2 node:20
enter data value in 3 node:30
enter data value in 4 node:40
enter data value in 5 node:50
 The list after creation:
10->20->30->40->50->head
 The list after deletion at end:
10->20->30->40->head


=== Code Execution Successful ===
```

6. **Aim:** Write a C program to delete a node after a given node(middle case) in a Double linked list.

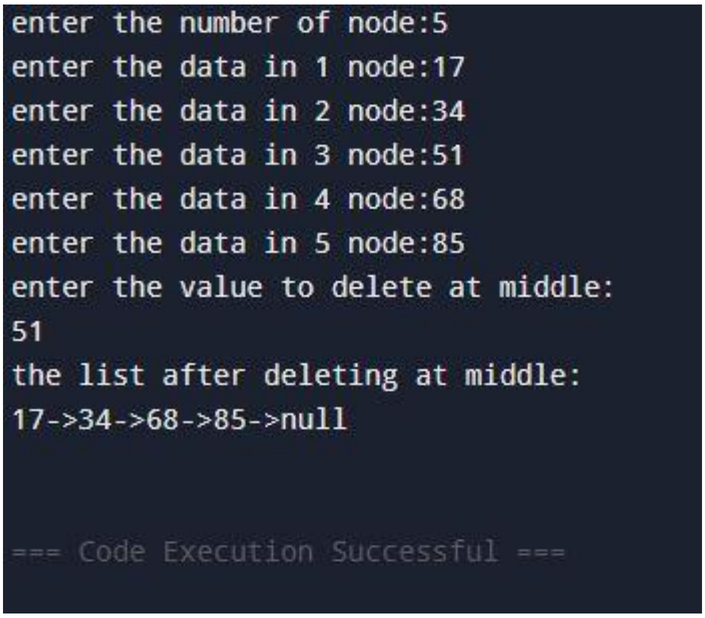
Program:

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node *next;

struct node *prev;

};

struct node *head,*temp,*new,*last=NULL;

struct node *getnode(int x)

{

struct node *temp=(struct node*)malloc(sizeof(struct node));

temp->data=x;

temp->next=NULL;

temp->prev=NULL;

}

void delete_middle(int val)

{

if(head==NULL)

 {

   printf("list is empty");

 }
```

```
else if(head->next==NULL)
{
 temp=head;
head=last=NULL;
free(temp);
}
else
{
temp=head;
while(temp!=NULL&&temp->data!=val)
{
temp=temp->next;
}
if(temp==NULL)
{
printf("node with value %d not found \n",val);
}
(temp->prev)->next=temp->next;
(temp->next)->prev=temp->prev;
free(temp);
}
}
void display()
{
if(head==NULL)
{
printf("list empty\n");
}
```

```
else
{
temp=head;
while(temp!=NULL)
{
printf("%d->",temp->data);
temp=temp->next;
}
}
printf("null\n");
}
int main()
{
int n,i,x,val;
printf("enter the number of node:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter the data in %d node:",i+1);
scanf("%d",&x);
new=getnode(x);
if(head==NULL)
{
head=last=new;
}
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=new;
new->prev=temp;
last=new;
last->next=NULL;
}
}
printf("enter the value to delete at middle:\n");
scanf("%d",&val);
printf("the list after deleting at middle:\n");
delete_middle(val);
display();
return 0;
}
```

Output:



```
enter the number of node:5
enter the data in 1 node:17
enter the data in 2 node:34
enter the data in 3 node:51
enter the data in 4 node:68
enter the data in 5 node:85
enter the value to delete at middle:
51
the list after deleting at middle:
17->34->68->85->null



=== Code Execution Successful ===
```

Inferences:

- Each node contains **three fields** → `data`, `prev` pointer, and `next` pointer.
- Traversal is possible in **both directions** (forward and backward).
- **Insertion and deletion** are easier compared to singly list (can be done from both ends efficiently).
- **More memory required** per node (extra `prev` pointer).
- Searching is still **O(n)** in worst case.
- **Head node's `prev` = NULL** and **last node's `next` = NULL**.
- Losing head pointer still causes list inaccessibility, but tail pointer helps in reverse traversals.
- Better suited for applications where **bidirectional traversal** is required (like undo/redo in editors, navigation systems).
- Compared to singly linked list, it provides **more flexibility**, but at the cost of **extra memory** and **slightly more complex operations**.