NAME: TANISHA.C.A

ROLL NO: 230701390

EX-2: Implementation of Double Linked List

```
#include<stdio.h>
#include<stdlib.h>
 void insert_beg(int); void
insert_end(int); void
insert_mid(int,int); void
display(); void del_beg();
void del_end(); void
del_mid(int); void
search(int); int count();
struct node {     int data;
struct node *prev,*next;
}*first=NULL,*last=NULL;

void insert_beg(int roll)
{
    struct node *newnode;
    newnode=(struct node *)malloc(sizeof(struct node));
newnode->data=roll;     if(first!=NULL){
newnode->prev=NULL;         newnode->next=first;
first->prev=newnode;         first=newnode;
    }
else{
         newnode->prev=NULL;
newnode->next=NULL;
first=newnode;          last=newnode;
        }
}
void insert_end(int roll)
{
    struct node *newnode;
    newnode=(struct node *)malloc(sizeof(struct node));
newnode->data=roll;     if(first==NULL)
    {
        newnode->prev=NULL;
        newnode->next=NULL;
first=newnode;         last=newnode;
    }     else     {
newnode->next=NULL;
newnode->prev=last;
last->next=newnode;
last=newnode;
    }
}
```

```
void insert_mid(int pos,int roll)
{
    struct node *newnode,*temp=first;
int c=count();
    newnode=(struct node *)malloc(sizeof(struct node));
newnode->data=roll;     if(pos==1)     {
        insert_beg(roll);
    }     else
if(pos>(c+1)){
        printf("\nOut of bounds\n");
    }     else
if(pos==c+1){
insert_end(roll);
    }
else     {
    for(int i=1;i<pos-1;i++)
    {
        temp=temp->next;
    }     newnode->next=temp-
>next;     newnode->prev=temp;
if(temp->next!=NULL){
    (temp->next)->prev=newnode;
    }
    temp->next=newnode;
    }
}
void display() {     struct
node *temp=NULL;
temp=first;
if(temp!=NULL){
while(temp!=NULL)
    {         printf("%d ",temp-
>data);         temp=temp->next;
    } } else{     printf("\nNo
data inside");
}
}
void del_beg() {     struct
node *temp=first;
first=temp->next;
free(temp);     first-
>prev=NULL;
    printf("\nDisplay after deleting first node\n");
display();
}
void del_end() {
    struct node *temp=first,*temp1=NULL;
while(temp->next!=NULL){
temp1=temp;         temp=temp->next;
    }     temp1-
>next=NULL;
free(temp);
    printf("\nDisplaying after deleting last node\n");
display();
```

```
}  int count() {     int
count=0;     struct node
*temp=first;
while(temp!=NULL)
    {
        temp=temp->next;
count++;         }
return count;
}
void del_mid(int pos)
{     if(pos==1){
del_beg();     }
    struct node *temp=first,*temp1=NULL;
for(int i=1;i<pos;i++){
temp1=temp;         temp=temp->next;
    }     temp1->next=temp-
>next;     (temp->next)-
>prev=temp1;     free(temp);
temp=NULL;
    printf("\nDisplay after deletion : ");
display();
}  void search(int
data)
{     int
c=1;
struct node
*temp=first;
if(first==NU
LL){
        printf("\nThe list is empty\n");
    }
else{
    while(temp!=NULL && temp->data!=data){
temp=temp->next;         c++;     }
if(c>count()){
    printf("\nNo data in list");
} else
    printf("\n%d is the position of data\n",c);
}
}
void del_all() {
    struct node *temp=first,*temp1=NULL;
while(temp!=NULL){         temp1=temp;
temp=temp->next;         free(temp1);
first=NULL;
    }
    temp=NULL;temp1=NULL;
    printf("\nAll data deleted successfully");
}
  int main() {     int n,ch,pos,t;
printf("MENU DRIVEN PROGRAM:\n");
printf("0. Exit\n");
    printf("1. Insert a node at the beginning\n");
printf("2. Insert a node at the end\n");
printf("3. Insert a node at any position\n");
printf("4. Search an element\n");     printf("5.
Delete at beginning \n");     printf("6. Delete at
any position\n");     printf("7. Delete at
end\n");     printf("8. Delete list\n");
printf("9. Display\n");     while(1){
    printf("\nEnter your choice :  ");
scanf("%d",&ch);     switch (ch)     {
case 1:
    printf("\nEnter roll to insert at beginning : ");
scanf("%d",&n);     insert_beg(n);     break;
case 2:
    printf("\nEnter roll to insert at end : ");
scanf("%d",&n);     insert_end(n);     break;      case
3:
    printf("Enter pos to insert : ");
scanf("%d",&pos);
    printf("\nEnter data to insert after pos : ");
scanf("%d",&n);     insert_mid(pos,n);     break;
case 4:
    printf("\nEnter data to search : ");
scanf("%d",&n);     search(n);
break;      case 5:     del_beg();
break;      case 6:
    printf("\nEnter pos to del : ");
scanf("%d",&pos);     del_mid(pos);
break;          case 7:
del_end();     break;      case 8:
del_all();     break;      case 9:
display();     break;
default:         printf("\nMENU
EXITED");         break;     }
if(ch==0){         break;     }
else      continue;
    } }
```

OUTPUT

```
1.Insert Beg

2.Insert Middle

3.Insert End

4.Delete Beg

5.Delete Middle

6.Delete End

7.Find

8.Traverse

9.Exit
```

```
Enter your choice : 1

Enter the element : 40

Enter your choice : 1

Enter the element : 30

Enter your choice : 1

Enter the element : 20

Enter your choice : 1

Enter the element : 10

Enter your choice : 8

10 20 30 40

Enter your choice : 7

Enter the element : 30 Element

found...!

Enter your choice : 1

Enter the element : 5

Enter your choice : 8

5 10 20 30 40

Enter your choice : 3

Enter the element : 45

Enter your choice : 8

5 10 20 30 40 45

Enter your choice : 2

Enter the position element : 20 Enter the element : 25

Enter your choice : 8

5 10 20 25 30 40 45

Enter your choice : 4

The deleted item is 5

Enter your choice : 8
```

```
10 20 25 30 40 45
```

```
Enter your choice : 6

The deleted item is 45

Enter your choice : 8

10 20 25 30 40

Enter your choice : 5

Enter the element : 30

The deleted item is 30

Enter your choice : 8

10 20 25 40

Enter your choice : 9
```