```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct NodeType
{
int info;
struct NodeType *next,*prev;
}Node;
/*function Declaration*/
void createList(Node **,Node **,int);
void displayList(Node *);
void removeList(Node **,Node **);
void InsertAtTop(Node **,Node **,int);
void InsertAtBottom(Node **,Node **,int);
void DeleteFromBottom(Node**,Node **);
void DeleteFromTop(Node **,Node **);
int count(Node*);
Node* search(Node*,int);
/*Function Definition*/
void createList(Node **head,Node ** tail,int n)
{
Node *toadd,*foradd;
int i,data;
for(i=0;i<n;i++)
{
toadd=(Node*)malloc(sizeof(Node));
if(toadd==(Node*)NULL)
{
printf("\nMemory not available:");
```

```
return;
}
printf("\nEnter the information in the list:\t");
scanf("%i",&data);
toadd->info=data;
toadd->next=(Node *)NULL;
if((*head)==(Node*)NULL)
{
toadd->prev=(Node*)NULL;
(*head)=(*tail)=toadd;
}
else
{
(*tail)->next=toadd;
toadd->prev=(*tail);
(*tail)=toadd;
}
}
}
void InsertAtTop(Node **top,Node **bottom,int data)
{
Node *toadd=(Node*)malloc(sizeof(Node));
int i;
if(toadd==(Node*)NULL)
{
printf("\nMemory not available:");
return;
}
toadd->info=data;
```

```
toadd->prev=(Node *)NULL;
if((*top)==(Node*)NULL)/*inserting first node at the top*/
{
toadd->next=(Node*)NULL;
(*top)=(*bottom)=toadd;
}
else
{
toadd->next=(*top);
(*top)->prev=toadd;
(*top)=toadd;
}
}
void InsertAtBottom(Node **head,Node **tail,int data)
{
Node *toadd,*foradd;
toadd=(Node*)malloc(sizeof(Node));
if(toadd==(Node*)NULL)
{
printf("\nMemory not available:");
return;
}
toadd->info=data;
toadd->next=(Node *)NULL;
if((*head)==(Node*)NULL)
{
toadd->prev=(Node*)NULL;
(*head)=(*tail)=toadd;
}
```

```
else
{
toadd->prev=(*tail);
(*tail)->next=toadd;
(*tail)=toadd;
}
}
void displayList(Node *disp)
{
if(disp==(Node*)NULL)
{
printf("\nNo Node present");
return;
}
while(disp!=(Node*)NULL)
{
printf("\t%d\t",disp->info);
disp=disp->next;
}
}
void removeList(Node **h,Node **t)
{
Node *del=*h;
while(del!=(Node*)NULL)
{
printf("\n Deleting node containing %d",del->info);
(*h)=del->next;
free(del);
del=(*h);
```

```
}
(*h)=(*t)=(Node *)NULL;
}
void DeleteFromTop(Node **h,Node **t)
{
Node *del=*h;
if((*h)==(Node*)NULL)
{
printf("\nThere is no node present in DLL");
return;
}
printf("\n Deleting node containing %d",del->info);
(*h)=del->next;
free(del);
if((*h)==(Node*)NULL)
(*h)=(*t)=(Node*)NULL;
}
void DeleteFromBottom(Node** head,Node **tail)
{
Node *todel,*fordel;
todel=(*tail);
if(todel==(Node*)NULL)
{
printf("\nOperation failed due to unvailability of node");
return;
}
if(todel==(*head))
{
(*head)=(*tail)=(Node*)NULL;
```

```
printf("\n%d is being now deleted",todel->info);
free(todel);
}
else
{
(*tail)=todel->prev;/*Making previous node as end node*/
todel->prev->next=(Node*)NULL;
printf("\n%d is being now deleted",todel->info);
free(todel);
}
}
int count(Node* tc)
{
int tot=0;
while(tc!=(Node*)NULL)
{
tot++;
tc=tc->next;
}
return tot;
}
Node* search(Node* ts,int element)
{
while(ts!=(Node*)NULL)
{
if(ts->info==element)
break;
else
ts=ts->next;
```

```
}
return ts;
}
/*Calling functions declared and defined above for execution*/
void main()
{
Node *beg,*end;
int hmany,ch;
beg=end=(Node*)NULL; /*Creating Empty SLL*/
do
{
clrscr();
printf("\nFollowing can be done:\n");
printf("\n1.\tCreate defined sized list");
printf("\n2.\tDisplay list elements");
printf("\n3.\tRemove Entire list");
printf("\n4.\tInsert at Top or beginning");
printf("\n5.\tDelete From Top");
printf("\n6.\tInsert at bottom");
printf("\n7.\tDelete from bottom");
printf("\n8.\tCount the elements");
printf("\n9.\tSearch an element");
printf("\n10.\tExit");
printf("\n?.\tEnter what you want to do:\t");
scanf("%i",&ch);
switch(ch)
{
case 1:printf("\nHow many elements you want to add:\t");
        scanf("%i",&hmany);
```

```
            createList(&beg,&end,hmany);
            break;
case 2:displayList(beg);
            break;
case 3:removeList(&beg,&end);
            break;
case 4:{int data;
            printf("\nEnter data to insert/place/push at top:\t");
            scanf("%i",&data);
            InsertAtTop(&beg,&end,data);
            break;}
case 5:printf("\nDeleting or pop element from top:\n");
            DeleteFromTop(&beg,&end);
            break;
case 6:
            {
            int data;
            printf("\nEnter element to insert/push/enqueue at Bottom:\t");
            scanf("%i",&data);
            InsertAtBottom(&beg,&end,data);
            break;}
case 7: printf("\nDeleting or pop or dequeue element from bottom:\n");
            DeleteFromBottom(&beg,&end);
            break;
case 8: printf("Total number of nodes present=%d",count(beg));
            break;
case 9:
            {
            Node *found;
```

```
            int data;
            printf("\nEnter the data to search in list:\t");
            scanf("%i",&data);
            found=search(beg,data);
            if(found==(Node*)NULL)
            printf("\nElement not found");
            else
            printf("\nElement %d found",found->info);
            break;
            }

case 10:if(beg!=(Node*)NULL)
            removeList(&beg,&end);
            else
            printf("\nYou may have removed the list");
            displayList(beg);
            return;
default:printf("\nOh! Something went wrong:\t%c",1);
            break;
}
printf("\nPress any key to continue...");
getch();
}while(ch!=10);
};
```