Program

```
/* DLL add:end delete:end*/
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

struct node
{
int data;
struct node *forward;
struct node *back;
};

struct headnode
{
int count;
struct node *pos;
struct node *head;
}*pList;

struct node *pPrev,*pLoc;


void printList( )
{
int i;

pList->pos=pList->head;
for (i=1;i<=pList->count;i++)
{
                printf("%d\t",pList->pos->data);
                pList->pos=pList->pos->forward;
}
}
```

```
void deleteNode( )
{
if (pPrev!=NULL)
{
        pPrev->forward=pLoc->forward;

        if (pLoc->forward!=NULL)
                pLoc->forward->back=pPrev;
}
else
{
        pList->head=pLoc->forward;
        if (pLoc->forward!=NULL)
                pLoc->forward->back=NULL;

}

pList->count =pList->count - 1;
free(pLoc);
}


void removeNode( )
{
        int i;
if (pList->count!=0){
        pPrev=NULL;
        pLoc=pList->head;
        for(i=1;i<pList->count;i++)
        {
                pPrev=pLoc;
                pLoc=pLoc->forward;
        }
        deleteNode( );
}
else
        printf("Error: No data\n");
}
```

```
int insertNode( int dataIn)
{
struct node *pNew;
pNew = (struct node *) malloc(sizeof(struct node));
if (pNew != NULL)
{       pNew->data=dataIn;
        if (pPrev!=NULL)
        {
                pNew->back=pPrev;
                pNew->forward=pPrev->forward;
                if (pPrev->forward!=NULL)/*OR if (pNew->forward!=NULL)*/
                        pPrev->forward->back=pNew;
                pPrev->forward=pNew;
        }
        else
        {       pNew->back=NULL;
                pNew->forward=pList->head;
                if (pNew->forward!=NULL) pList->head->back=pNew;
                pList->head=pNew;
        }
        pList->count +=1;
        return 1;
}
else    return 0;
}

void addNode( int dataIn)
{
        int i,success;
        pPrev=NULL;
        pLoc=pList->head;
        for(i=1;i<=pList->count;i++)
        {
                pPrev=pLoc;
                pLoc=pLoc->forward;
        }
        success=insertNode(dataIn);
        if (success)  printf("Data Inserted Successfully\n");
        else          printf("Out of Memory.....\n");
}
```

```
 int menu( )
{
int choice;
printf("\n\n********************\n\n");
printf(" ....   M E N U ...\n");
printf("1: Add end\n");
printf("2: Delete end\n");
printf("3: Print List\n");
printf("4: Quit\n\n");
printf("********************\n\n");

printf("feed in your choice: ");
scanf("%d",&choice);

return choice;
}


void createList( )
{
pList = (struct headnode *)malloc(sizeof(struct headnode));
if (pList != NULL)
{
pList->count=0;
pList->head = NULL;
}
else
{
        printf("Insufficient memory...\n");
        exit(0);
}
}
```

```
void main( )
{
int choice;
int dataIn,deleteKey;

createList( );

do
{
        choice = menu();

        if (choice==1)
    {
                printf("Feed in the data: ");
                scanf("%d",&dataIn);
                addNode(dataIn);
    }
        else
        if (choice==2)
        {
                removeNode();
        }
        else
        if (choice==3)
            printList( );

} while(choice!=4);

}
```

HOME OF EDUCATION

Navlakhi®

www.navlakhi.com
Home of Education

DATA Structures@ Navlakhi™