LAB PROGRAM 8: Doubly Liked List

```
/*****************************************************************************
Develop a menu driven Program in C for the following operations on Doubly Linked List (DLL) of
Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo
a. Create a DLL of N Employees Data by using end insertion.
b. Display the status of DLL and count the number of nodes in it
c. Perform Insertion and Deletion at End of DLL
d. Perform Insertion and Deletion at Front of DLL
e. Demonstrate DLL as double ended queue
f. Exit
*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Enode
{
char ssn[15];
char name[20];
char dept[5];
char designation[10];
int salary;
long long int phno;
struct Enode *left;
struct Enode *right;
}*head=NULL;
struct Enode *tail,*temp1,*temp2;
void create(char [],char [],char [],char [],int ,long long int);
void ins_beg(char [],char [],char [],char [],int ,long long int);
void ins_end(char [],char [],char [],char [],int ,long long int);
void del_beg();
void del_end();
void display();

void main()
{
 int choice;
 char s[15],n[20],dpt[5],des[10];
 int sal;
 long long int p;
 printf("1.Create\n2.Display\n3.Insert at beginning\n4.Insert at End\n5.Delete at begining\n
          6.Delete at End\n7.Exit\n");
 while(1)
 {
  printf("\nEnter your choice\n");
  scanf("%d",&choice);
  switch(choice)
  {
    case 1: printf("Enter the required data(Emp no,Name,Dept,Desig,sal,phone\n");
            scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);
            create(s,n,dpt,des,sal,p);
```

```
                break;
        case 2: display();
                 break;
        case 3: printf("Enter the required data (Emp no,Name,Dept,Desig,sal,phone\n");
                scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);
                ins_beg(s,n,dpt,des,sal,p);
                break;
        case 4: printf("Enter the required data(Emp no,Name,Dept,Desig,sal,phone\n");
                scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);
                ins_end(s,n,dpt,des,sal,p);
                break;
        case 5: del_beg();
                 break;
        case 6: del_end();
                 break;
        case 7: exit(0);
       }
    }
}

void create(char s[15],char n[20],char dpt[5],char des[10],int sal,long long int p)
{
          temp1=(struct Enode *)malloc(1*sizeof(struct Enode));
          strcpy(temp1->ssn,s);
          strcpy(temp1->name,n);
          strcpy(temp1->dept,dpt);
          strcpy(temp1->designation,des);
          temp1->salary=sal;
          temp1->phno=p;
          temp1->left=NULL;
          temp1->right=NULL;
         if(head==NULL)
         {
             head=tail=temp1;
          }
        else
        {
         tail->right=temp1;
         temp1->right=NULL;
         temp1->left=tail;
         tail=temp1;
         }
   }

void display()
{
          int count=0;
          temp1=head;
          if(head==NULL)
          {
```

```
        printf("empty list\n");
        return;
        }
        printf("Employee Details .....\n");
        while(temp1!=NULL)
        {
        printf("------------------\n");
        printf("SSN:%s\nNAME:%s\nDEPT:%s\nDESIGN:%s\nSALARY:%d\nPHN:
%lld\n",temp1->ssn,temp1->name,temp1->dept,temp1->designation,temp1->salary,temp1->phno);
        printf("------------------\n");
        temp1=temp1->right;
        count++;
        }
        printf("Number of nodes=%d\n",count);
}

void ins_beg(char s[15],char n[20],char dpt[5],char des[10],int sal,long long int p)
{
temp1=(struct Enode *)malloc(1*sizeof(struct Enode));
strcpy(temp1->ssn,s);
strcpy(temp1->name,n);
strcpy(temp1->dept,dpt);
strcpy(temp1->designation,des);
temp1->salary=sal;
temp1->phno=p;
temp1->right=head;
head->left=temp1;
head=temp1;
temp1->left=NULL;
}

void ins_end(char s[15],char n[20],char dpt[5],char des[10],int sal,long long int p)
{
temp1=(struct Enode *)malloc(1*sizeof(struct Enode));
strcpy(temp1->ssn,s);
strcpy(temp1->name,n);
strcpy(temp1->dept,dpt);
strcpy(temp1->designation,des);
temp1->salary=sal;
temp1->phno=p;
tail->right=temp1;
temp1->left=tail;
temp1->right=NULL;
tail=temp1;
}

void del_beg()
{
  if(head==NULL)
  {
```

```
   printf("List is empty\n"); //DLL does not contain any node
  }
 else
 {
   temp1=head->right;
   if(temp1!=NULL)        //DLL has many nodes
   {
     temp1->left=NULL;
     printf("deleted node is:\n");
     printf("----------------------\n");
     printf("SSN:%s\nNAME:%s\nDEPT:%s\nDESIGN:%s\nSALARY:%d\nPHN:%lld\n",head-
            >ssn,head->name,head->dept,head->designation,head->salary,head->phno);
    printf("----------------------\n");
     free(head);
     head=temp1;
   }
   else
    {      //DLL has only one node, that is head
     printf("deleted node is:\n");
     printf("----------------------\n");
     printf("SSN:%s\nNAME:%s\nDEPT:%s\nDESIGN:%s\nSALARY:%d\nPHN:%lld\n",head-
             >ssn,head->name,head->dept,head->designation,head->salary,head->phno);
     printf("----------------------\n");
     free(head);
     head=NULL;
    }
  }
}

void del_end()
{
  if(head==NULL) // DLL is empty
   {
    printf("empty list\n");
   }
   else
  {
   temp1=tail;
   if(head==tail) //only one node
   {
      printf("deleted node is:\n");
      printf("----------------------\n");
      printf("SSN:%s\nNAME:%s\nDEPT:%s\nDESIGN:%s\nSALARY:%d\nPHN:%lld\n",tail-
              >ssn,tail->name,tail->dept,tail->designation,tail->salary,tail->phno);
      printf("----------------------\n");
      head=tail=NULL;
      free(temp1);
    }
    else     //many nodes in DLL
    {
```

```
        printf("deleted node is:\n");
        printf("----------------------\n");
        printf("SSN:%s\nNAME:%s\nDEPT:%s\nDESIGN:%s\nSALARY:%d\nPHN:%lld\n",tail-
                >ssn,tail->name,tail->dept,tail->designation,tail->salary,tail->phno);
        printf("----------------------\n");
        tail=temp1->left;
        tail->right=NULL;
        free(temp1);
    }
  }
}
```

e. Demonstration of DLL Double ended Queue:

Dqueue or double ended queue is a generalized version of queue that allows insert and delete at both ends.

Operations on Deque :

insertFront() : Adds an item at the front of Deque. ==> ins_beg( ) of DLL
insertRear() : Adds an item at the rear of Deque. ==> ins_end( ) of DLL
deleteFront() : Deletes an item from front of Deque. ==> del_beg( ) of DLL
deleteRear() : Deletes an item from rear of Deque. ==> del_end( ) of DLL