**ASSIGNMENT 3 > PROGRAM 1: DOUBLY LINKED LIST**

*Source Code –*

/\*Doubly Linked List

Author: Asmit De \*/

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

typedef struct node

{

int data;

struct node \*llink, \*rlink;

}NODE;

NODE \*getNode(int data)

{

NODE \*node = (NODE\*)calloc(1,sizeof(NODE));

node->data = data;

node->llink = node->rlink = NULL;

return node;

}

void returnNode(NODE \*node)

{

free(node);

}

NODE \*addBegin(NODE \*list,char ch,int data)

{

NODE \*node = getNode(data);

if(ch == 'l' || ch == 'L')

{

node->rlink = list;

if(list != NULL)

list->llink = node;

}

else

{

node->llink = list;

if(list != NULL)

list->rlink = node;

}

return node;

}

int addAfterPos(NODE \*list,int pos,int data)

{

NODE \*node;

int i=1;

if(list == NULL)

return 1;

while(i<pos)

{

if(list->rlink != NULL)

{

list = list->rlink;

i++;

}

else

return 1;

}

node = getNode(data);

node->rlink = list->rlink;

node->llink = list;

list->rlink = node;

if(node->rlink != NULL)

node->rlink->llink = node;

return 0;

}

int addAfterData(NODE \*list,int edata,int data)

{

NODE \*node;

if(list == NULL)

return 1;

while(list != NULL && list->data != edata)

{

list = list->rlink;

if(list == NULL)

return 1;

}

node = getNode(data);

node->rlink = list->rlink;

node->llink = list;

list->rlink = node;

if(node->rlink != NULL)

node->rlink->llink = node;

return 0;

}

NODE \*delBegin(NODE \*list,char ch)

{

NODE \*node = list;

if(ch == 'l' || ch == 'L')

list = list->rlink;

else

list = list->llink;

returnNode(node);

return list;

}

int delPos(NODE \*list,int pos)

{

NODE \*header = list;

int i=1;

if(list == NULL)

return 1;

else if(pos == 1)

{

header = delBegin(list,'l');

return 0;

}

else

{

while(i<pos)

{

if(list->rlink != NULL)

{

list = list->rlink;

i++;

}

else

return 1;

}

}

if(list->llink != NULL)

list->llink->rlink = list->rlink;

if(list->rlink != NULL)

list->rlink->llink = list->llink;

returnNode(list);

return 0;

}

int delData(NODE \*list,int edata)

{

NODE \*header = list;

if(list == NULL)

return 1;

else if(list->data == edata)

{

header = delBegin(list,'l');

return 0;

}

else if(list->rlink == NULL)

return 1;

else

{

while(list->data != edata)

{

if(list->rlink != NULL)

list = list->rlink;

else

return 1;

}

}

if(list->llink != NULL)

list->llink->rlink = list->rlink;

if(list->rlink != NULL)

list->rlink->llink = list->llink;

returnNode(list);

return 0;

}

void displayList(NODE \*list,char ch)

{

system("cls");

printf("List: ");

if(list != NULL)

do

{

printf("%d->",list->data);

if(ch == 'l' || ch == 'L')

list = list->rlink;

else

list = list->llink;

}while(list != NULL);

printf("NULL");

}

void main()

{

char ch;

int data,edata,pos,flag;

NODE \*lheader = NULL, \*rheader = NULL;

while(1)

{

system("cls");

puts("MENU");

puts("Key\tFunction");

puts("1\tDisplay list");

puts("2\tAdd node at beginning of list");

puts("3\tAdd node at end of list");

puts("4\tAdd node after a position in list");

puts("5\tAdd node after an existing data in list");

puts("6\tDelete node from beginning of list");

puts("7\tDelete node from end of list");

puts("8\tDelete node at a position in list");

puts("9\tDelete node containing an existing data in list");

puts("x\tExit");

puts("\nEnter your choice...");

ch=getch();

flag=0;

switch(ch)

{

case '1':

system("cls");

do

{

printf("\nEnter l to start from left or r to start from right: ");

ch=getche();

}while(!(ch == 'l' || ch == 'r' || ch == 'L' || ch == 'R'));

if(ch == 'l' || ch == 'L')

displayList(lheader,ch);

else

displayList(rheader,ch);

puts("\nPress any key to return to menu...");

getch();

break;

case '2':

system("cls");

printf("\nEnter data to add: ");

scanf("%d",&data);

if(lheader == NULL && rheader == NULL)

lheader = rheader = addBegin(NULL,'l',data);

else

lheader = addBegin(lheader,'l',data);

printf("\nData added successfully...\nPress any key to return to menu...");

getch();

break;

case '3':

system("cls");

printf("\nEnter data to add: ");

scanf("%d",&data);

if(lheader == NULL && rheader == NULL)

lheader = rheader = addBegin(NULL,'l',data);

else

rheader = addBegin(rheader,'r',data);

printf("\nData added successfully...\nPress any key to return to menu...");

getch();

break;

case '4':

system("cls");

printf("\nEnter data to add: ");

scanf("%d",&data);

printf("Enter the position after which the data is to be added: ");

scanf("%d",&pos);

if(!addAfterPos(lheader,pos,data))

printf("\nData added successfully...\nPress any key to return to menu...");

else

printf("\nError: Invalid position number...\nPress any key to return to menu...");

getch();

break;

case '5':

system("cls");

printf("Enter data to add: ");

scanf("%d",&data);

printf("Enter the existing data after which the data is to be added: ");

scanf("%d",&edata);

if(!addAfterData(lheader,edata,data))

printf("\nData added successfully...\nPress any key to return to menu...");

else

printf("\nError: Given data not found...\nPress any key to return to menu...");

getch();

break;

case '6':

system("cls");

if(lheader != NULL)

{

lheader = delBegin(lheader,'l');

printf("\nNode deleted successfully...\nPress any key to return to menu...");

}

else

printf("\nError: List is empty...\nPress any key to return to menu...");

getch();

break;

case '7':

system("cls");

if(rheader != NULL)

{

rheader = delBegin(rheader,'r');

printf("\nNode deleted successfully...\nPress any key to return to menu...");

}

else

printf("\nError: List is empty...\nPress any key to return to menu...");

getch();

break;

case '8':

system("cls");

printf("Enter the position of node which is to be deleted: ");

scanf("%d",&pos);

if(!delPos(lheader,pos))

printf("\nNode deleted successfully...\nPress any key to return to menu...");

else

printf("\nError: Invalid position number...\nPress any key to return to menu...");

getch();

break;

case '9':

system("cls");

printf("Enter the data which is to be deleted: ");

scanf("%d",&edata);

if(!delData(lheader,edata))

printf("\nNode deleted successfully...\nPress any key to return to menu...");

else

printf("\nError: Given data not found...\nPress any key to return to menu...");

getch();

break;

case 'x':

flag=1;

break;

default:

system("cls");

puts("Invalid input...\nPress any key to return to menu...");

getch();

break;

}

if(flag)

break;

}

}