WEEK-06

DOUBLY LINKED LIST

#include<stdio.h>

#include<stdlib.h>

struct NODE{

int data;

struct NODE \*Llink,\*Rlink;

};

typedef struct NODE node;

node \*start=NULL,\*new,\*curr,\*temp;

void create();

void display();

void insert();

void delete();

void main()

{

int choice;

while(1)

{

printf("1.Create \n 2.Display \n 3.Insert \n 4.Delete the specific value \n 5.Exit \n");

printf("Enter the choice:");

scanf("%d",&choice);

switch(choice)

{

case 1: create();

break;

case 2: display();

break;

case 3: insert();

break;

case 4: delete();

break;

case 5: exit(0); break;

default:printf("Wrong Choice");

}

}

getch();

}

void create()

{

int ch;

start=(node\*) malloc(sizeof(node));

curr=start;

printf("Enter an element");

scanf("%d",&start->data);

while(1)

{

printf("Do you want to enter a new element (1 for yes,any other number for no)");

scanf("%d",&ch);

if(ch==1)

{

new=(node\*) malloc(sizeof(node));

curr->Rlink=new;

new->Llink=curr;

printf("Enter an element");

scanf("%d",&new->data);

curr=new;

}

else

{

curr->Rlink=NULL;

break;

}

}

}

void display()

{

if(start==NULL)

{

printf("Linked List is empty");

return;

}

temp=start;

while(temp!=NULL)

{

printf("%d\n",temp->data);

temp=temp->Rlink;

}

}

void insert()

{

new=(node\*)malloc(sizeof(node));

printf("Enter an element");

scanf("%d",&new->data);

if(start==NULL)

{

new->Llink=NULL;

new->Rlink=NULL;

start=new;

return;

}

start->Llink=new;

new->Rlink=start;

new->Llink=NULL;

start=new;

}

void delete()

{

int ele;

if(start==NULL)

{

printf("Linked list is empty");

return;

}

printf("Enter the element to be deleted:");

scanf("%d",&ele);

if(start->Rlink!=NULL&&start->data==ele)

{

temp=start;

start=start->Rlink;

start->Llink=NULL;

free(temp);

return;

}

temp=start;

while(temp->Rlink!=NULL&&temp->data!=ele)

{

temp=temp->Rlink;

}

if(temp->data&&temp->Rlink==NULL)

{

temp->Llink->Rlink=NULL;

free(temp);

return;

}

if(temp->data==ele&&temp->Rlink!=NULL)

{

temp->Llink->Rlink=temp->Rlink;

temp->Rlink->Llink=temp->Llink;

free(temp);

return;

}

}

OUTPUT:

