**Program No. 7**

Objective:-WAP for various operation performed on Doubly Linked list

#include<stdio.h>

#include<stdlib.h>

struct node\*head=NULL;

struct node

{

int data;

struct node\*llink;

struct node\*rlink;

};

struct node\*Aakash\_createnewnode()

{

struct node \*newNode;

newNode=(struct node\*) malloc (sizeof(struct node));

printf("\nEnter new node=\n");

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

newNode->llink=NULL;

newNode->rlink=NULL;

return newNode;

}

void printLinkedlist(struct node \*ptr)

{

while(ptr->rlink!=NULL)

{

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

ptr=ptr->rlink;

}printf("%d",ptr->data);

}

void createdoublylinkedlistbyaddingnodeatfront()

{

struct node\*newNode,\*ptr;

if(head==NULL)

{

printf("NO! node in the linked list=\n");

head=Aakash\_createnewnode();

}

else

{

newNode=Aakash\_createnewnode();

newNode->rlink=head;

head->llink=newNode;

head=newNode;

}

}

void createdoublylinkedlistbyaddingnodeatend()

{

struct node\*newNode,\*ptr;

if(head==NULL)

{

printf("NO! node in the linked list=\n");

head=Aakash\_createnewnode();

}

else

{

newNode=Aakash\_createnewnode();

ptr=head;

while(ptr->rlink!=NULL)

{

ptr=ptr->rlink;

}

newNode->llink=ptr;

ptr->rlink=newNode;

}

}

void createdoublylinkedlistbyaddingnodeatanyposition(int key)

{

struct node\*newNode,\*ptr;

if(head==NULL)

{

printf("NO! node in the linked list=\n");

head=Aakash\_createnewnode();

}

else

{

ptr=head;

printLinkedlist(ptr);

while(ptr->data!=key)

{

ptr=ptr->rlink;

}

if(ptr->rlink!=NULL)

{

newNode=Aakash\_createnewnode();

newNode->llink=ptr;

newNode->rlink=ptr->rlink;

ptr->rlink=newNode;

printLinkedlist(head);

}

else

{

newNode=Aakash\_createnewnode();

newNode->llink=ptr;

ptr->rlink=newNode;

printLinkedlist(head);

}

}

}

void deletedoublylinkedlistbyremovingnodefromfront()

{

struct node \*ptr;

if(head==NULL)

{

printf("No! There is no node");

exit(1);

}

else if(head->rlink==NULL)

{

ptr=head;

free(ptr);

printf("No! There is no node");

exit(1);

}

else

{

ptr=head;

head=head->rlink;

head->llink=NULL;

free(ptr);

}

}

void deletedoublylinkedlistbyremovingnodefromend()

{

struct node \*ptr, \*ptr1;

if(head==NULL)

{

printf("No! There is no node");

exit(1);

}

else if(head->rlink==NULL)

{

ptr=head;

free(ptr);

printf("No! There is no node");

exit(1);

}

else

{

ptr=head;

while(ptr->rlink!=NULL)

{

ptr=ptr->rlink;

}

ptr1=ptr->llink;

ptr1->rlink=NULL;

free(ptr);

}

}

void deletedoublylinkedlistbyremovingnodefromanyposition(int key)

{

struct node \*ptr, \*ptr1,\*ptr2;

ptr=head;

while(ptr->data!=key)

{

ptr=ptr->rlink;

}

if(ptr->rlink!=NULL)

{

ptr1=ptr->llink;

ptr2=ptr->rlink;

ptr1->rlink=ptr2;

ptr2->llink=ptr1;

free(ptr);

}

else if(ptr->rlink==NULL)

{

ptr1=ptr->llink;

ptr1->rlink=NULL;

free(ptr);

}

else if(ptr==NULL)

{

ptr=head;

head=head->rlink;

free(ptr);

}

}

void main()

{

int n;

char choice;struct node \*ptr;int key;

printf("Enter the value between (1-6)=\n");

printf("\n\nFor Creating Doubly Linked List By Adding Node At Front Press 1:");

printf("\n\nFor Creating Doubly Linked List By Adding Node At End Press 2:");

printf("\n\nFor Creating Doubly Linked List By Adding Node At Any Position Press 3:");

printf("\n\nFor Deleting Any Node From Doubly Linked List By Front Press 4:");

printf("\n\nFor Deleting Any Node From Doubly Linked List By End Press 5:");

printf("\n\nFor Deleting Any Node From Doubly Linked List By Any Position Press 6:\n\n");

scanf("%d",&n);

printf("\n");

switch(n)

{

case 1:

{

do

{

createdoublylinkedlistbyaddingnodeatfront();

printf("Do u want to add further node=\n");

choice=getch();

}

while(choice=='Y'||choice=='y');

ptr=head;

printf("Your Linked List is in order of=\n");

printLinkedlist(ptr);

getch();

break;

}

case 2:

{

do

{

createdoublylinkedlistbyaddingnodeatend();

printf("Do u want to add further node=\n");

choice=getch();

}

while(choice=='Y'||choice=='y');

ptr=head;

printf("Your Linked List is in order of=\n");

printLinkedlist(ptr);

getch();

break;

}

case 3:

{

do

{

createdoublylinkedlistbyaddingnodeatanyposition(key);

printf("\nEnter the value of key=\n");

scanf("%d",&key);

printf("Do u want to add further node=\n");

choice=getch();

}

while(choice=='Y'||choice=='y');

printf("Your Linked List is in order of=\n");

printLinkedlist(head);

getch();

break;

}

case 4:

{

do

{

createdoublylinkedlistbyaddingnodeatend();

ptr=head;

printLinkedlist(ptr);

printf("\nDo u want to add further node=\n");

choice=getch();

}

while(choice=='Y'||choice=='y');

ptr=head;

printf("\nYour Linked List is in order of=\n");

printLinkedlist(ptr);

printf("\n\nYour Deleted Linked List Is=\n");

do

{

deletedoublylinkedlistbyremovingnodefromfront();

ptr=head;

printf("\n");

printLinkedlist(ptr);

printf("\n\nDo u want to delete further node=\n");

choice=getch();

}

while(choice=='Y'||choice=='y');

ptr=head;

printf("\n\nYour Linked List is in order of=\n");

printLinkedlist(ptr);

break;

}

case 5:

{

do

{

createdoublylinkedlistbyaddingnodeatend();

ptr=head;

printLinkedlist(ptr);

printf("\nDo u want to add further node=\n");

choice=getch();

}

while(choice=='Y'||choice=='y');

ptr=head;

printf("\nYour Linked List is in order of=\n");

printLinkedlist(ptr);

printf("\n\nYour Deleted Linked List Is=\n");

do

{

deletedoublylinkedlistbyremovingnodefromend();

ptr=head;

printf("\n");

printLinkedlist(ptr);

printf("\n\nDo u want to delete further node=\n");

choice=getch();

}

while(choice=='Y'||choice=='y');

ptr=head;

printf("\n\nYour Linked List is in order of=\n");

printLinkedlist(ptr);

break;

}

case 6:

{

do

{

createdoublylinkedlistbyaddingnodeatend();

printf("Do u want to add further node=\n\n");

choice=getch();

}

while(choice=='Y'||choice=='y');

ptr=head;

printf("Your Linked List is in order of=\n\n");

printLinkedlist(head);

printf("\nEnter the key=\n");

scanf("%d",&key);

deletedoublylinkedlistbyremovingnodefromanyposition(key);

puts("\nLinked List after Deletion of key:\n\n");

printLinkedlist(head);

getch();

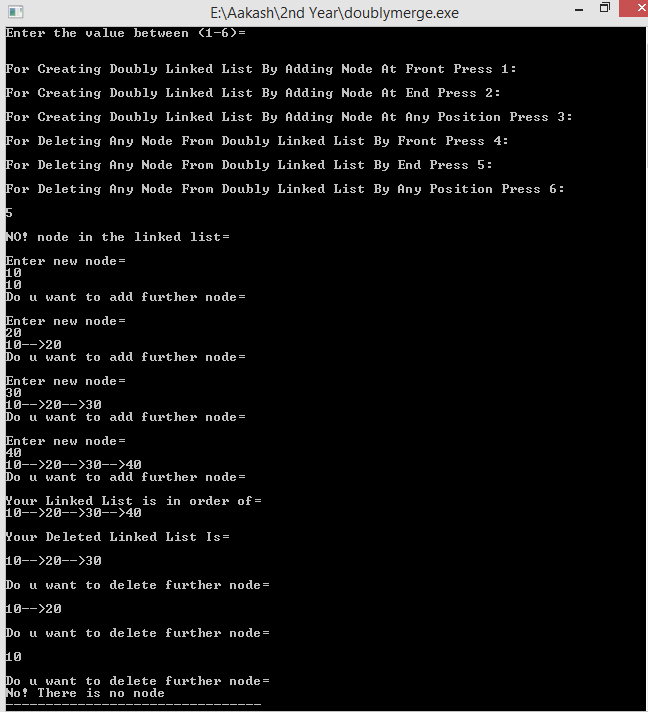
break;

}

}

}

Output:-



Operation performed on Doubly Linked list