**Program No. 8**

Objective:-WAP for various operation performed on Doubly Circular 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=newNode;

newNode->rlink=newNode;

return newNode;

}

void printLinkedlist(struct node \*ptr)

{

ptr=head;

while(ptr->rlink!=head)

{

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

ptr=ptr->rlink;

}

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

}

void createdoublycircularlinkedlistbyaddingnodeatfront()

{

struct node\*newNode,\*ptr;

if(head==NULL)

{

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

head=Aakash\_createnewnode();

head->rlink=head->llink=head;

}

else

{

ptr=head;

newNode=Aakash\_createnewnode();

while(ptr->rlink!=head)

{

ptr=ptr->rlink;

}

newNode->rlink=head;

head->llink=newNode;

newNode->llink=ptr;

ptr->rlink=newNode;

head=newNode;

}

}

void createdoublycircularlinkedlistbyaddingnodeatend()

{

struct node\*newNode,\*ptr;

if(head==NULL)

{

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

head=Aakash\_createnewnode();

head->rlink=head->llink=head;

}

else

{

ptr=head;

newNode=Aakash\_createnewnode();

while(ptr->rlink!=head)

{

ptr=ptr->rlink;

}

ptr->rlink=newNode;

newNode->llink=ptr;

newNode->rlink=head;

head->llink=newNode;

}

}

void createdoublycircularlinkedlistbyaddingnodeatanyposition(int key)

{

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

if(head==NULL)

{

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

head=Aakash\_createnewnode();

head->rlink=head->llink=head;

}

else if(head->rlink==head)

{

ptr=head;

while((ptr->data!=key)&&(ptr->rlink!=head))

{

ptr=ptr->rlink;

}

if(ptr->rlink!=NULL)

{

newNode=Aakash\_createnewnode();

newNode->llink=ptr;

ptr->rlink=newNode;

newNode->rlink=head;

}

}

else

{

ptr=head;

while(ptr->data!=key)

{

ptr=ptr->rlink;

}

if(ptr->rlink!=head)

{

ptr1=ptr->rlink;

newNode=Aakash\_createnewnode();

newNode->llink=ptr1;

ptr1->rlink=newNode;

ptr->rlink=newNode;

}

else

{

newNode=Aakash\_createnewnode();

while(ptr->rlink!=head)

{

ptr=ptr->rlink;

}

ptr->rlink=newNode;

newNode->llink=ptr;

newNode->rlink=head;

head->llink=newNode;

}

}

}

void deletedoublycircularlinkedlistbyremovingnodefromfront()

{

struct node \*ptr,\*ptr1;

if(head==NULL)

{

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

exit(1);

}

else if(head->rlink==head)

{

ptr=head;

free(ptr);

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

exit(1);

}

else

{

ptr=head;ptr1=head;

while(ptr->rlink!=head)

{

ptr=ptr->rlink;

}

head=head->rlink;

ptr->rlink=head;

head->llink=ptr;

free(ptr1);

}

}

void deletedoublycircularlinkedlistbyremovingnodefromend()

{

struct node \*ptr, \*ptr1;

if(head==NULL)

{

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

exit(1);

}

else if(head->rlink==head)

{

ptr=head;

free(ptr);

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

exit(1);

}

else

{

ptr=head;

while(ptr->rlink!=head)

{

ptr=ptr->rlink;

}

ptr1=ptr->llink;

ptr1->rlink=head;

free(ptr);

}

}

void deletedoublycircularlinkedlistbyremovingnodefromanyposition(int key)

{

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

ptr=head;

while(ptr->data!=key)

{

ptr=ptr->rlink;

}

if(ptr->rlink!=head)

{

ptr1=ptr->llink;

ptr2=ptr->rlink;

ptr1->rlink=ptr2;

ptr2->llink=ptr1;

free(ptr);

}

else if(ptr->rlink==head)

{

ptr1=ptr->llink;

ptr1->rlink=head;

head->llink=ptr1;

free(ptr);

}

else if(ptr->rlink==head)

{

free(ptr);

printf("Now! there is no node found in the list");

exit(1);

}

}

void main()

{

int n,m,o;

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

printf("Enter the value between (1-2)=\n");

printf("\n\nFor Creating Circular Doubly Linked List Press 1:");

printf("\n\nFor Deleting Node From Circular Doubly Linked List Press 2:\n\n");

scanf("%d",&n);

printf("\n");

switch(n)

{

case 1:

{

printf("Enter the value between (1-3)=\n");

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

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

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

scanf("%d",&m);

printf("\n");

switch(m)

{

case 1:

{

do

{

createdoublycircularlinkedlistbyaddingnodeatfront();

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

choice=getch();

}

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

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

printLinkedlist(head);

getch();

break;

}

case 2:

{

do

{

createdoublycircularlinkedlistbyaddingnodeatend();

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(ptr);

getch();

break;

}

case 3:

{

do

{

createdoublycircularlinkedlistbyaddingnodeatanyposition(key);

printLinkedlist(ptr);

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

scanf("%d",&key);

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

choice=getch();

}

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

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

printLinkedlist(head);

getch();

break;

}

}

break;

}

case 2:

{

printf("Enter the value between (1-3)=\n");

printf("\n\nFor Deleting Any Node From Circular Doubly Linked List By Front Press 1:");

printf("\n\nFor Deleting Any Node From Circular Doubly Linked List By End Press 2:");

printf("\n\nFor Deleting Any Node From Circular Doubly Linked List By Any Position Press 3:\n\n");

scanf("%d",&o);

printf("\n");

switch(o)

{

case 1:

{

do

{

createdoublycircularlinkedlistbyaddingnodeatend();

ptr=head;

printLinkedlist(ptr);

printf("\n\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

{

deletedoublycircularlinkedlistbyremovingnodefromfront();

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 2:

{

do

{

createdoublycircularlinkedlistbyaddingnodeatend();

ptr=head;

printLinkedlist(ptr);

printf("\n\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

{

deletedoublycircularlinkedlistbyremovingnodefromend();

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 3:

{

do

{

createdoublycircularlinkedlistbyaddingnodeatend();

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);

deletedoublycircularlinkedlistbyremovingnodefromanyposition(key);

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

printLinkedlist(head);

getch();

break;

}

break;

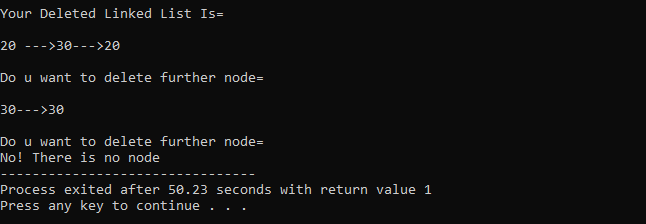
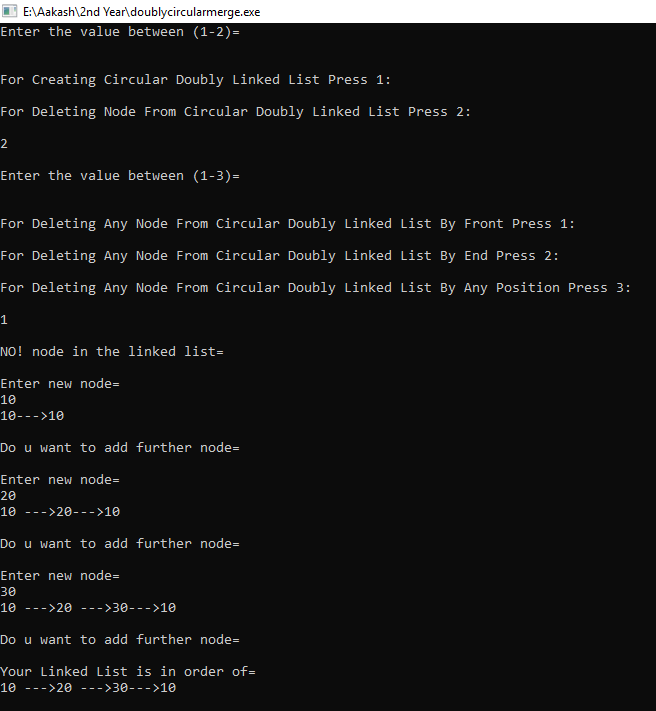
}

}break;

}

}

Output:-



Operation performed on Doubly Circular Linked list