***// QUEUE USING DOUBLY LINKED LIST***

#include<stdlib.h>

#include<stdio.h>

struct node

{

int data;

struct node \*next,\*prev;

};

struct dequeue

{

struct node \*front;

struct node \*rear;

};

void qdisplay(struct dequeue);

void qinsert\_tail(int ,struct dequeue\*);

int qdelete\_head(struct dequeue\*);

int main()

{

int k,x,ch;

struct dequeue dq;

dq.front=dq.rear=NULL;

while(1)

{

qdisplay(dq);

printf("\n1..insert head");

printf("\n2..insert Tail");

printf("\n3..Delete Head");

printf("\n4..Delete Tail");

printf("\n5..display");

printf("\n6..EXIT");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("Enter the value..");

scanf("%d",&x);

qinsert\_head(x,&dq);

break;

case 2:printf("Enter the value..");

scanf("%d",&x);

qinsert\_tail(x,&dq);

break;

case 3:k=qdelete\_head(&dq);

printf("Element deleted = %d\n",k);

break;

case 4:k=qdelete\_tail(&dq);

printf("Element deleted = %d\n",k);

break;

case 5:qdisplay(dq);

}

}

}

void qdisplay(struct dequeue dq)

{

struct node \*p,\*q;

p=dq.front;

q=dq.rear;

if(p==NULL)

printf("Empty queue\n");

else

{

while(p!=q)

{

printf("%d<->",p->data);

p=p->next;

}

printf("%d<->",p->data);

}

}

void qinsert\_tail(int x ,struct dequeue \*dq)

{

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

temp->data=x;

temp->prev=NULL;

temp->next=NULL;

if(dq->front==NULL) //first node

{

dq->front=temp;

dq->rear=temp;

}

else

{

dq->rear->next=temp; //insert at end

temp->prev=dq->rear;

dq->rear=temp;

}

}

int qdelete\_head(struct dequeue \*dq)

{

struct node \*q;

int x;

if(dq->front==NULL) //empty queue

return -1;

q=dq->front;

x=q->data;

if(dq->front==dq->rear)

dq->front=dq->rear=NULL;

else

{

dq->front=q->next; //dq->front->next

dq->front->prev=NULL;

}

free(q);

return x;

}