#include<stdio.h>

#include<stdlib.h>

struct Priority

{

int data;

int priority;

};

struct Que

{

struct Priority\* arr;

int front;

int size;

int capacity;

};

void create\_que(struct Que\*\* que)

{

(\*que) = (struct Que\*)malloc(sizeof(struct Que));

if(\*que == NULL)

{

printf("No memory allocated\n");

}

else

{

printf("Enter capacity\n");

scanf("%d",&((\*que)->capacity));

(\*que)->arr= (struct Priority\*)malloc(((\*que)->capacity)\*sizeof(struct Priority));

(\*que)->front=0;

(\*que)->size=0;

}

}

int isFull(struct Que\* que)

{

if((que->capacity)==(que->size))

{

return 1;

}

else return 0;

}

int isEmpty(struct Que\* que)

{

if((que->size)==0)

{

return 1;

}

else return 0;

}

/\*

void insert\_first(struct Que\* que,int element,int prio)

{

if(isFull(que))

{

printf("Que is Full\n");

}

else

{

(que->front) = ((que->front) + (que->capacity) - 1) % (que->capacity);

que->arr[que->front].data = element;

que->arr[que->front].priority = prio;

(que->size)++;

}

}

\*/

void insert\_first(struct Que\* que,int element,int prio)

{

if(isFull(que))

{

printf("Que is Full\n");

}

else

{

que->front = ((que->front)+ (que->capacity)-1) % (que->capacity);

que->arr[que->front].data = element;

que->arr[que->front].priority = prio;

(que->size)=(que->size)+1;

printf("inserted %d at front\n",que->arr[que->front].data);

}

}

void delet(struct Que\* que)

{

int maxPrio=que->front;

int i;

int j = que->front;

if(isEmpty(que))

{

printf("Que is empty\n");

}

else

{

for(i=1; i<=que->size; i++)

{

if((que->arr[j].priority)<(que->arr[maxPrio].priority))

{

j=(j+1)%(que->capacity);

}

else if((que->arr[j].priority)>(que->arr[maxPrio].priority))

{

maxPrio=j;

j=(j+1)%(que->capacity);

}

else

{

if((que->arr[j].data)>(que->arr[maxPrio].data))

{

maxPrio = j;

j=(j+1)%(que->capacity);

}

else

{

j = (j+1)%(que->capacity);

}

}

}

printf("Deleted %d from que\n",que->arr[maxPrio].data);

for(i=maxPrio; i<(((que->front)+(que->size))%(que->capacity)); i++)

{

que->arr[i].data = que->arr[i+1].data;

que->arr[i].priority = que->arr[i+1].priority;

}

(que->size)--;

}

}

void insert\_rear(struct Que\* que,int element,int prio)

{

if(isFull(que))

{

printf("Deque is Full\n");

}

else

{

que->arr[((que->front)+(que->size))%(que->capacity)].data=element;

que->arr[((que->front)+(que->size))%(que->capacity)].priority=prio;

printf("inserted %d at rear\n",que->arr[((que->front) + (que->size)) % (que->capacity)]);

(que->size)++;

}

}

void display(struct Que\* que)

{

if(isEmpty(que))

{

printf("Queue is Empty\n");

}

else

{

int m,i;

m = que->front;

for(i=1; i<=que->size; i++)

{

printf("%d->",que->arr[m].data);

m = (m+1)%(que->capacity);

}

printf("\n");

}

}

void main()

{

struct Que\* que = NULL;

int ch;

int element;

int prio;

create\_que(&que);

do

{

printf("Enter your choice:\n1.insert\_first\n2.delete\n3.insert\_rear\n5.Display\n0.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("Enter value to insert\n");

scanf("%d",&element);

printf("Enter priority\n");

scanf("%d",&prio);

insert\_first(que,element,prio);

break;

case 2: delet(que);

break;

case 3: printf("Enter element to insert at last\n");

scanf("%d",&element);

printf("Enter priority\n");

scanf("%d",&prio);

insert\_rear(que,element,prio);

break;

/\*

case 4: delete\_last(que);

break;

\*/

case 5: display(que);

break;

case 0: exit(0);

break;

default: printf("Please enter valied choice\n");

}

}while(ch!=0);

}