/\*WAP to simulate the working of a **queue** of integers using an array. Provide the following operations a) Insert b) Delete c) Display D)to peek. The program should print appropriate messages for queue empty and queue overflow conditions

\*/

/\*

#include <stdio.h>

#define size 5

int arr[size];

int rare=-1;

int front=-1;

void Display();

void Enqueue();

void Dequeue();

void peek();

void main()

{

int choice;

while(1)

{

printf("1.Enqueue\n");

printf("2.Dequeue\n");

printf("3.Display\n");

printf("4.Peek\n");

printf("5.Exit \n");

printf("enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:Enqueue();

break;

case 2:Dequeue();

break;

case 3:Display();

break;

case 4:peek();

break;

case 5:exit(1);

default:printf("Invalid input\n");

}

}

}

void peek()

{

if(front==-1)

printf("Queue underflow\n");

else

printf("The peek element is=%d\n ",arr[front]);

}

void Display()

{

int i;

if(front==-1)

printf("Queue is Empty\n");

else{

printf("Queue elements:\n");

for(i=front;i<=rare;i++)

{

printf("%d",arr[i]);

printf("\n");

}

}

}

void Dequeue()

{

if(front ==-1||front>rare)

{

printf("Queue underflow\n");

return;

}

else{

printf("Deleted element is:%d\n",arr[front]);

front=front+1;

}

}

void Enqueue()

{

int item;

if(rare==(size-1))

printf("Queue Overflow\n");

else

{

if(front ==-1)

front=0;

printf("Enter the element to be inserted\n");

scanf("%d",&item);

rare=rare+1;

arr[rare]=item;

}

}

/\*

WAP to simulate the working of **a circular queue** of integers using an array.

Provide the following operations. a) Insert b) Delete c) Display.

\*/

/\*

#include <stdio.h>

#include<stdlib.h>

#define size 5

int Q[size];

int rear=-1;

int front=-1;

int IsFull()

{

if(front==(rear+1)%size)

{

return 0;

}

else

{

return -1;

}

}

int IsEmpty()

{

if(front==-1 && rear==-1)

{

return 0;

}

else

{

return -1;

}

}

void Enqueue(int x)

{

int item;

if(IsFull()==0)

{

printf("Queue Overflow\n");

return;

}

else if(IsEmpty()==0)

{

front=0;

rear=0;

}

else

{

rear=(rear+1)%size;

}

Q[rear]=x;

}

int Dequeue()

{

int x;

if(IsEmpty()==0)

{

printf("Queue underflow\n");

}

else if(front==rear)

{

x=Q[front];

front=-1;

rear=-1;

}

else

{

x=Q[front];

front=(front+1)%size;

}

return x;

}

void Display()

{

int i;

if(IsEmpty()==0)

printf("Queue is Empty\n");

else

{

printf("Queue elements:\n");

for(i=front;i!=rear;i=((i+1)%size))

{

printf("%d\n",Q[i]);

}

printf("%d\n",Q[i]);

}

}

void main()

{

int choice,x,b;

while(1)

{

printf("1.Enqueue,2.Dequeue,3.Display,4.Exit\n");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("Enter the number to be inserted into the queue\n");

scanf("%d",&x);

Enqueue(x);

break;

case 2:b=Dequeue();

printf("%d was removed from the queue\n",b);

break;

case 3:Display();

break;

case 4:exit(1);

default:printf("Invalid input\n");

}

}

}

\*/

/\* **dequeue**

#include<stdio.h>

#include<stdlib.h>

#define qsize 5

int f=0,r=-1,ch;

int item,q[10];

int isfull()

{

return(r==qsize-1)?1:0;

}

int isempty()

{

return(f>r)?1:0;

}

void insert\_rear()

{

if(isfull())

   {

   printf("queue overflow\n");

   return;

   }

r=r+1;

q[r]=item;

}

void delete\_front()

{

if(isempty())

   {

   printf("queue empty\n");

   return;

   }

printf("item deleted is %d\n",q[(f)++]);

if(f>r)

   {

   f=0;

   r=-1;

   }

}

void insert\_front()

{

if(f!=0)

   {

   f=f-1;

   q[f]=item;

   return;

   }

  else if((f==0)&&(r==-1))

   {

   q[++(r)]=item;

   return;

   }

  else

   printf("insertion not possible\n");

}

void delete\_rear()

{

if(isempty())

   {

   printf("queue is empty\n");

   return;

   }

printf("item deleted is %d\n",q[(r)--]);

if(f>r)

   {

   f=0;

   r=-1;

   }

}

void display()

{

int i;

if(isempty())

   {

   printf("queue empty\n");

   return;

   }

for(i=f;i<=r;i++)

  printf("%d\n",q[i]);

}

void main()

{

for(;;)

{

  printf("1.insert\_rear\n2.insert\_front\n3.delete\_rear\n4.delete\_front\n5.display\n6.exit\n");

  printf("enter choice\n");

  scanf("%d",&ch);

  switch(ch)

   {

   case 1:printf("enter the item\n");

       scanf("%d",&item);

       insert\_rear();

       break;

   case 2:printf("enter the item\n");

       scanf("%d",&item);

       insert\_front();

       break;

   case 3:delete\_rear();

       break;

   case 4:delete\_front();

       break;

   case 5:display();

       break;

   default:exit(0);

   }

  }

}

\*/

/\*

#include <stdio.h>

#include <stdlib.h>

#define q\_size 5

int r=-1,f=0,item,count=0;

int q[10],ch;

void insert\_rear(){

  if (r==q\_size-1){

    printf("Queue overflow\n");

    return;

  }

  r=r+1;

  q[r]=item;

  count++;

}

void insertion\_sort(){

  int i,j,key;

  for (i=1;i<count;i++)

  {

    key=q[i];

    j=i-1;

    while (j>=0 && q[j]>key){

      q[j+1]=q[j];

      j=j-1;

    }

    q[j+1]=key;

  }

}

void delete\_front(){

  if (f>r){

    f=0;

    r=-1;

    printf("Queue is empty\n");

    return;

  }

  printf("Item deleted=%d\n",q[f++]);

}

void display(){

  if (f>r){

    printf("Queue is empty\n");

    return;

  }

  printf("Contents of the queue are:\n");

  for(int i=f;i<=r;i++)

  {

      printf("%d\n",q[i]);

  }

}

int main(){

  for (;;)

  {

    printf("\n1:insert\_rear\n2:delete\_front\n3:display\n");

    printf("Enter the choice:\n");

    scanf("%d",&ch);

    switch (ch){

      case 1:printf("Enter the item:\n");

       scanf("%d",&item);

       insert\_rear();

       insertion\_sort();

       break;

     case 2:delete\_front();

     break;

     case 3:display();

     break;

     default:exit(0);

    }

  }

  return 0;

}

\*/

/\*

#include <stdio.h>

#include <stdlib.h>

#define q\_size 5

int r=-1,f=0,item,count=0;

int q[10],ch;

void insert\_rear(){

  if (r==q\_size-1){

    printf("Queue overflow\n");

    return;

  }

  r=r+1;

  q[r]=item;

  count++;

}

void insertion\_sort(){

  int i,j,key;

  for (i=1;i<count;i++)

  {

    key=q[i];

    j=i-1;

    while (j>=0 && q[j]>key){

      q[j+1]=q[j];

      j=j-1;

    }

    q[j+1]=key;

  }

}

void delete\_rear(){

  if (f>r){

    f=0;

    r=-1;

    printf("Queue is empty\n");

    return;

  }

  printf("Item deleted=%d\n",q[r--]);

}

void display(){

  if (f>r){

    printf("Queue is empty\n");

    return;

  }

  printf("Contents of the queue are:\n");

  for(int i=f;i<=r;i++)

  {

      printf("%d\n",q[i]);

  }

}

int main(){

  for (;;)

  {

    printf("\n1:insert\_rear\n2:delete\_front\n3:display\n");

    printf("Enter the choice:\n");

    scanf("%d",&ch);

    switch (ch){

      case 1:printf("Enter the item:\n");

       scanf("%d",&item);

       insert\_rear();

       insertion\_sort();

       break;

     case 2:delete\_rear();

     break;

     case 3:display();

     break;

     default:exit(0);

    }

  }

  return 0;

}

\*/

**// 4) Priority queue**

#include<stdio.h>

#include<stdlib.h>

#define size 5

int pri\_Q[size];

int rear=-1;

int front=-1;

void Display();

void check(int);

void Priority\_insertion(int);

void delete\_front();

void main()

{

int choice,ch;

while(1)

{

printf("1.To Insert\n");

printf("2.To Delete\n");

printf("3.To Display\n");

printf("4.Exit \n");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("Enter the value to be inserted\n");

scanf("%d",&ch);

Priority\_insertion(ch);

break;

case 2:delete\_front();

break;

case 3:Display();

break;

case 4:exit(1);

break;

default:printf("Invalid input\n");

}

}

}

void Priority\_insertion(int item)

{

if(rear>=(size-1))

{

printf("Queue Overflow\n");

return;

}

else if(front ==-1 && rear==-1)

{

front++;

rear++;

pri\_Q[rear]=item;

return;

}

else

check(item);

rear++;

}

void check(int item)

{

int i,j;

//<= ascending priority queue

//>= descending priority queue

for(i=0;i<=rear;i++)

{

if(item<=pri\_Q[i])

{

for(j=rear+1;j>i;j--)

{

pri\_Q[j]=pri\_Q[j-1];

}

pri\_Q[i]=item;

return;

}

}

pri\_Q[i]=item; //in case that item is greater than all the elements

}

/\*

//for deleting a given value in priority queue

void Priority\_deletion(int item)

{

int i;

if(front==-1 && rear==-1)

{

printf("Queue underflow\n");

return;

}

else

{

for(i=0;i<=rear;i++)

{

if(item==pri\_Q[i])

{

while(i<rear)

{

pri\_Q[i]=pri\_Q[i+1];

i++;

}

pri\_Q[i]=-99; //What is the need of this line?

rear--;

if(rear==-1)

front=-1;;

return;

}

}

printf("%d element to be deleted not found",item);

}

}

\*/

void delete\_front() // Won’t it create error becoz display is printing from i=0 and also enqueue, dequeue are from i=0 not i=front; //What is the use of this line?



{

if(front==-1 && rear==-1)

{

printf("Queue empty\n");

return;

}

else

{

printf("Queue item deleted=%d\n",pri\_Q[front++]);

}

}

void Display()

{

if(front==-1 && rear==-1)

printf("Queue is Empty\n");

else

{

printf("Queue elements:\n");

while(front<=rear)

{

printf("%d",pri\_Q[front]);

printf("\n");

front++;

}

front=0; //Won’t the initial front position is distorted?

}

}