04/11/2020

**Experiment No: 10**

**PRIORITY QUEUE USING ARRAYS**

**AIM**:

Write a program to implement Priority Queue using arrays.

**DATA STRUCTURES USED:**

Queue

**ALGORITHM:**

Algorithm INSERT (ITEM, VALUE) // N is the size

1. If (REAR=N-1)
2. print “Queue is full”
3. Exit
4. Else
5. If (REAR =-1 && FRONT =-1)
6. FRONT=REAR=0
7. Queue[REAR]=ITEM
8. PRIORITY[REAR]=VALUE
9. Else
10. Queue[++REAR]=ITEM
11. PRIORITY[REAR]=VALUE
12. EndIf
13. EndIf

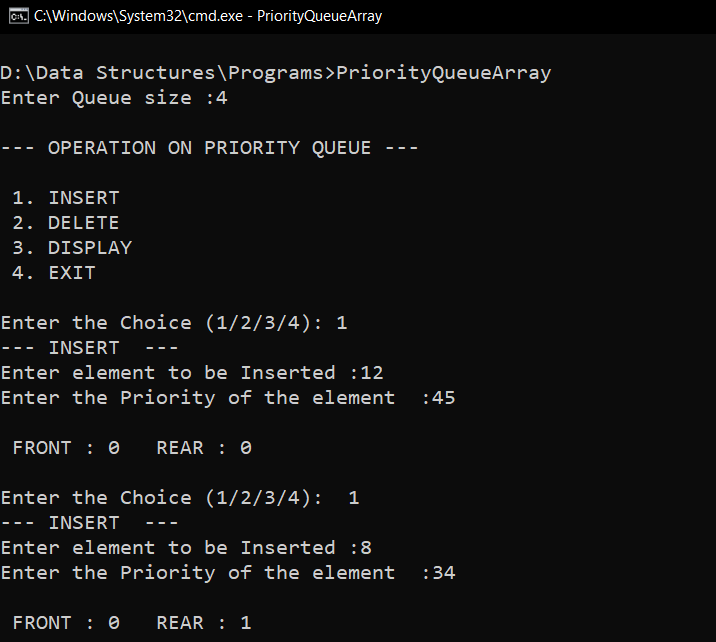
Algorithm DELETE

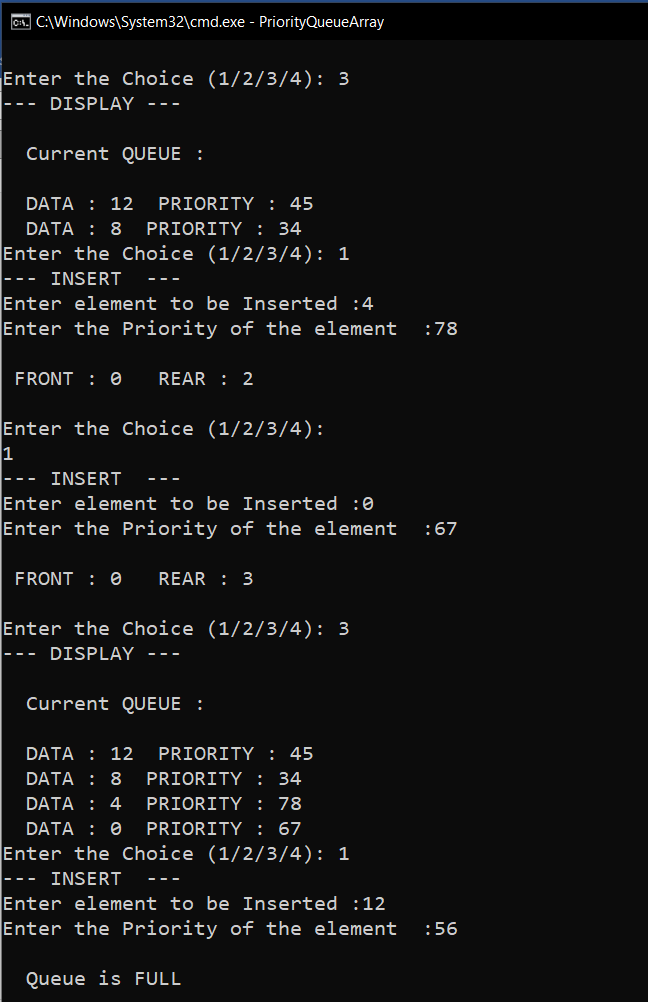
1. If (FRONT=-1)
2. Print “Queue is empty”
3. Exit
4. Else
5. If (FRONT == REAR)
6. ITEM = Queue[FRONT]
7. VALUE=PRIORITY[FRONT]
8. FRONT=REAR=-1
9. Else
10. SORT() // Sort According to the Priority(Descending Order)
11. VALUE=PRIORITY[FRONT]
12. ITEM = Queue[FRONT]
13. FRONT++
14. EndIf
15. EndIf

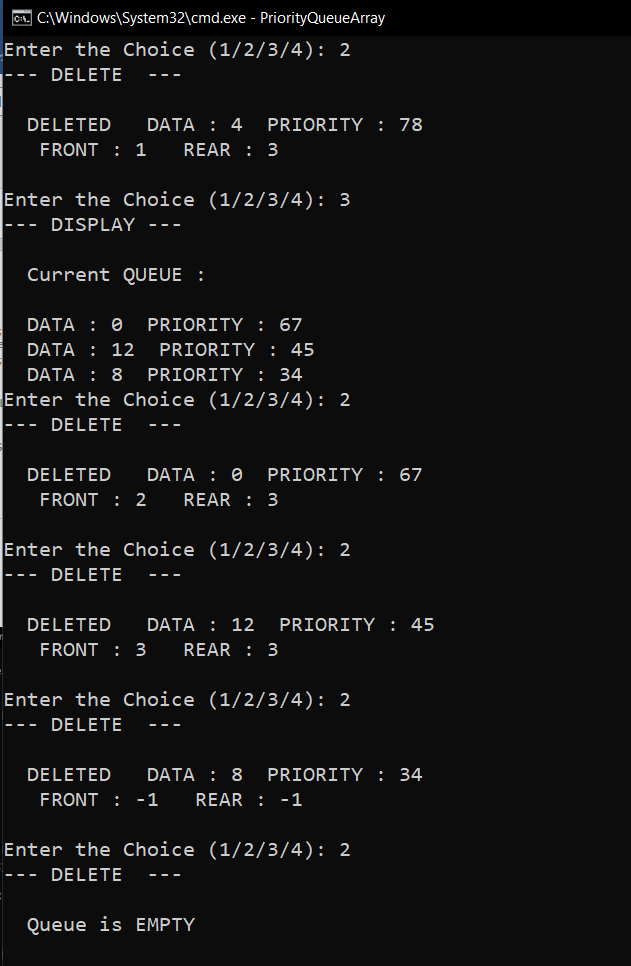
**PROGRAM:**

#include <stdio.h>  
#include <stdlib.h>  
int size;  
int front=-1;  
int rear=-1;  
int \*arr;  
int \*pty;  
  
void enqueue (int item,int priority){  
 if(rear==size-1){  
 printf("\n Queue is FULL\n\n");  
 }  
 else if(front==-1){  
 arr[++front] = item;  
 pty[front] = priority;  
 ++rear;  
 }  
 else {  
 arr[++rear] = item;  
 pty[rear] = priority;  
 }  
 printf("\n FRONT : %d REAR : %d \n",front,rear);  
  
}  
void dequeue(){  
 if(front==-1){  
 printf("\n Queue is EMPTY\n\n");  
 }  
 else if(front==rear){  
 int item = arr[front];  
 int priority = pty[front];  
 printf("\n DELETED DATA : %d PRIORITY : %d ",item,priority);  
 front=-1;  
 rear=-1;  
  
 }else{  
 //Sorting the Queue according to Priority  
 int i,j,n,temp1,temp2;  
 for (i=0 ; i<rear-front;i++){  
 for (j=0 ; j<rear-front-i;j++){  
 if (pty[j] < pty[j+1]){  
 temp1=pty[j];  
 pty[j]=pty[j+1];  
 pty[j+1]=temp1;  
 temp2=arr[j];  
 arr[j]=arr[j+1];  
 arr[j+1]=temp2;  
 }  
 }  
 }  
 int item = arr[front];  
 int priority = pty[front];  
 printf("\n DELETED DATA : %d PRIORITY : %d ",item,priority);  
 front++;  
 }  
 printf("\n FRONT : %d REAR : %d \n",front,rear);  
}  
  
void display(){  
 printf("\n Current QUEUE :\n");  
 if(front==-1){  
 printf("\n Queue is EMPTY \n");  
 }else{  
 for(int i=front; i<=rear; i++){  
 printf("\n DATA : %d PRIORITY : %d ",arr[i],pty[i]);  
 }  
 }  
}  
  
void main(){  
 int n,x,y;  
 char ans='y';  
 printf("Enter Queue size :");  
 scanf("%d", &size);  
 arr = (int\*) malloc (size \* sizeof(int));  
 pty = (int\*) malloc (size \* sizeof(int));  
 printf("\n--- OPERATION ON PRIORITY QUEUE --- \n\n");  
 printf(" 1. INSERT \n");  
 printf(" 2. DELETE\n");  
 printf(" 3. DISPLAY\n");  
 printf(" 4. EXIT\n");  
 while(ans=='y'){  
 printf("\nEnter the Choice (1/2/3/4): ");  
 scanf("%d",&n);  
 switch(n){  
 case 1:printf("--- INSERT ---\n");  
 printf("Enter element to be Inserted :");  
 scanf("%d", &x);  
 printf("Enter the Priority of the element :");  
 scanf("%d", &y);  
 enqueue(x,y);  
 break;  
 case 2:printf("--- DELETE ---\n");  
 dequeue();  
 break;  
 case 3:printf("--- DISPLAY ---\n");  
 display();  
 break;  
 case 4:ans='n';  
 break;  
 default:printf("Enter a Valid Input\n");  
 }  
 }  
}

**OUTPUT:**







**RESULT:**

The Priority Queue was successfully implemented and the required operations were carried out.

Time complexity of Insert() operation is O(1).

Time complexity of Delete() operation is O(n2).