

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define QueueLimit 34
5
6  typedef int QueueElementType;
7
8  typedef struct {
9      int Front, Rear;
10     QueueElementType Element[QueueLimit];
11 } QueueType;
12
13 typedef enum {FALSE, TRUE} boolean;
14
15 void CreateQ(QueueType *Queue);
16 void RemoveQ(QueueType *Queue, QueueElementType *Item);
17 void AddQ(QueueType *Queue, QueueElementType Item);
18 void TraverseQ(QueueType Queue);
19 boolean EmptyQ(QueueType Queue);
20 boolean FullQ(QueueType Queue);
21 boolean SearchQ(QueueType *Queue, QueueElementType Item);
22
23 int main()
24 {
25     QueueType myQueue;
26     QueueElementType i,searchItem;
27
28     CreateQ(&myQueue);
29
30     /*????????? ??? ??? ? ? ? ?????????? ??? 3.*/
31     for(i = 3; i < 100; i+=3)
32         AddQ(&myQueue,i);
33
34     /*????????? ??? ????.*/
35     TraverseQ(myQueue);
36
37     /*?????? ? ? ????????? ???? ??????????*/
38     printf("Give the search value:");
39     scanf("%d",&searchItem);
40
41     /*????????????? ??? ????????? SearchQ ?? ???????
42     ??? ??????? ??? ??? ? ? ????????? ???????
43     ????????? ??????.*/
44     if(SearchQ(&myQueue,searchItem))
45     {
46         printf("Found\n");
47         TraverseQ(myQueue);
48     }
49     else
50         printf("Item not Found");
51
52     return 0;
53 }
54
55 void CreateQ(QueueType *Queue)
56 {
57     Queue->Front = 0;
58     Queue->Rear = 0;
59 }
60
61 void RemoveQ(QueueType *Queue, QueueElementType *Item)
62 {
63     if(!EmptyQ(*Queue))
64     {
65         *Item = Queue ->Element[Queue -> Front];
66         Queue ->Front = (Queue ->Front + 1) % QueueLimit;

```

```

67     }
68     else
69         printf("Empty Queue");
70 }
71
72 void AddQ(QueueType *Queue, QueueElementType Item)
73 {
74     int NewRear;
75
76     if(!FullQ(*Queue))
77     {
78         NewRear = (Queue ->Rear + 1) % QueueLimit;
79         Queue ->Element[Queue ->Rear] = Item;
80         Queue ->Rear = NewRear;
81     }
82     else
83         printf("Full Queue");
84 }
85
86 void TraverseQ(QueueType Queue) {
87     int current;
88     current = Queue.Front;
89     while (current != Queue.Rear) {
90         printf("%d ", Queue.Element[current]);
91         current = (current + 1) % QueueLimit;
92     }
93     printf("\n");
94 }
95
96 boolean EmptyQ(QueueType Queue)
97 {
98     return (Queue.Front == Queue.Rear);
99 }
100
101 boolean FullQ(QueueType Queue)
102 {
103     return ((Queue.Front) == ((Queue.Rear +1) % QueueLimit));
104 }
105
106 boolean SearchQ(QueueType *Queue, QueueElementType Item)
107 {
108     /*? ????????? found ????? ????? flag ,?? ?????? ? ???? ???
109     ?????? ??? ??? ????????? ?????? ?????????? ?? ?????.*?
110     int current,found=0;
111     current = (*Queue).Front;
112
113     /*????????? ??? ?? ?????????? ??? ?????? ??? ?????? ,?????
114     ??? ?????? ??? ?? ? ?????????? ??? ?????????? ? ??????????????
115     ??? ?????? ??? ?? ?????? ?????? ??? ?????????? ? ???? ??? ?????????????
116     found ?????? 1 ,????????????? ??? ?????????? ??? ?????????????? ???
117     ?????? ?????????? ??? ???? ??? found.*?
118     while(current != (*Queue).Rear)
119     {
120         if(Queue ->Element[current] != Item)
121             RemoveQ(&(*Queue),&Queue ->Element[current]);
122         else
123         {
124             RemoveQ(&(*Queue),&Queue ->Element[current]);
125             found=1;
126             break;
127         }
128
129         current = (current + 1) % QueueLimit;
130     }
131
132     return found;

```

133
134 }
135