```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  typedef struct node
4.  {
5.  void*  dataPtr;
6.  struct node* next;
7.  } QUEUE_NODE;
8.  typedef struct
9.  {
10.         QUEUE_NODE* front;
11.         QUEUE_NODE* rear;
12.         int  count;
13.         } QUEUE;
14.         QUEUE* createQueue  (void);
15.         bool  enqueue   (QUEUE* queue, void*  itemPtr);
16.         void printQueue   (QUEUE* stack);
17.         int main (void)
18.         {
19.         QUEUE* queue1;
20.         QUEUE* queue2;
21.         QUEUE* queue3;
22.         int*   numPtr;
23.         int** itemPtr;
24.         queue1 = createQueue();
25.         queue2 = createQueue();
26.         queue3 = createQueue();



27.         int i=11;
28.         numPtr  = (int*)malloc(sizeof(i));
29.         *numPtr = i;
30.         enqueue(queue1, numPtr);
```

```
31.     i=22;
32.     numPtr  = (int*)malloc(sizeof(i));
33.     *numPtr = i;
34.     enqueue(queue1, numPtr);

35.     i=33;
36.     numPtr  = (int*)malloc(sizeof(i));
37.     *numPtr = i;
38.     enqueue(queue1, numPtr);

39.     i=44;
40.     numPtr  = (int*)malloc(sizeof(i));
41.     *numPtr = i;
42.     enqueue(queue2, numPtr);

43.     i=55;
44.     numPtr  = (int*)malloc(sizeof(i));
45.     *numPtr = i;
46.     enqueue(queue2, numPtr);

47.     i=66;
48.     numPtr  = (int*)malloc(sizeof(i));
49.     *numPtr = i;
50.     enqueue(queue2, numPtr);

51.     i=77;
52.     numPtr  = (int*)malloc(sizeof(i));
53.     *numPtr = i;
54.     enqueue(queue3, numPtr);
```

```
55.     i=88;
56.     numPtr  = (int*)malloc(sizeof(i));
57.     *numPtr = i;
58.     enqueue(queue3, numPtr);

59.     i=99;
60.     numPtr  = (int*)malloc(sizeof(i));
61.     *numPtr = i;
62.     enqueue(queue3, numPtr);

63.     printf ("Queue 1:\n");
64.     printQueue (queue1);
65.     printf ("Queue 2:\n");
66.     printQueue (queue2);
67.     printf ("Queue 3:\n");
68.     printQueue (queue3);
69.     return 0;
70.     }
71.     QUEUE* createQueue (void)
72.     {
73.     QUEUE* queue;
74.     queue = (QUEUE*) malloc (sizeof (QUEUE));
75.     if (queue)
76.     {
77.     queue->front  = NULL;
78.     queue->rear   = NULL;
79.     queue->count  = 0;
80.     }
81.     return queue;
82.     }
83.     bool enqueue (QUEUE* queue, void* itemPtr)
84.     {
85.     QUEUE_NODE* newPtr = (QUEUE_NODE*)malloc(sizeof(QUEUE_NODE));
```

```
86.     newPtr->dataPtr = itemPtr;
87.     newPtr->next = NULL;
88.     if (queue->count == 0)
89.     queue->front  = newPtr;
90.     else
91.     queue->rear->next = newPtr;
92.     (queue->count)++;
93.     queue->rear = newPtr;
94.     return true;
95.     }
96.     QUEUE* destroyQueue (QUEUE* queue)
97.     {
98.     QUEUE_NODE* deletePtr;
99.     if (queue)
100.    {
101.    while (queue->front != NULL)
102.    {
103.    free (queue->front->dataPtr);
104.    deletePtr   = queue->front;
105.    queue->front = queue->front->next;
106.    free (deletePtr);
107.    }
108.    free (queue);
109.    }
110.    return NULL;
111.    }
112.    void printQueue(QUEUE* queue)
113.    {
114.    QUEUE_NODE* node = queue->front;
115.    printf ("Front=>");
116.    while (node)
117.    {
118.    printf ("%3d", *(int*)node->dataPtr);
119.    node = node->next;
```

```
120.    }
121.    printf(" <=Rear\n");
122.    return;
123.    }
```