```c
/* LinkQueue.c */
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include <assert.h>
#include "LinkQueue.h"

//队列初始化
void QueueNew(QUEUE *q, int keySize, QueueFree *freeFn)
{
    assert(keySize > 0);
    q->keySize = keySize;
    q->head.prev = q->head.next = &(q->head);
    q->size = 0;
    q->freeFn = freeFn;
}

//队列销毁
void QueueDispose(QUEUE *q)
{
    QUEUENODE *cur, *post;
    for(cur = q->head.next; cur != &(q->head); cur = post)
    {
        post = cur->next;
        if(NULL != q->freeFn)
        {
            q->freeFn(cur->key);
        }
        free(cur);
    }
    q->head.next = q->head.prev = &(q->head);
    q->size = 0;
}

//入队操作，新节点插入到队尾
int QueueEn(QUEUE *q, const void *e)
{
    QUEUENODE *newNode = malloc(sizeof(QUEUENODE) + q->keySize);
    if(NULL == newNode)
    {
        return -1;
    }
    newNode->next = &(q->head);
    newNode->prev = q->head.prev;
    newNode->next->prev = newNode;
    newNode->prev->next = newNode;
    memcpy(newNode->key, e, q->keySize);
    q->size ++;
    return 0;
}

//队列判空
int QueueEmpty(QUEUE *q)
{
    return (0 == q->size);
}

//队列节点数量
int QueueSize(QUEUE *q)
{
    return q->size;
}
//出队操作，节点从队头出队
int QueueDe(QUEUE *q, void *e)
{
    if (QueueEmpty(q))
    {
        return -1;
    }
    QUEUENODE *node = q->head.next;
    memcpy(e, node->key, q->keySize);
    node->next->prev = node->prev;
    node->prev->next = node->next;
```

```
74      free(node);
75      q->size --;
76      return 0;
77  }
78
79  //获取队头元素
80  int QueueTop(QUEUE *q, void *e)
81  {
82      if (QueueEmpty(q))
83      {
84          return -1;
85      }
86      QUEUENODE *node = q->head.next;
87      memcpy(e, node->key, q->keySize);
88      return 0;
89  }
90
91  //获取队尾元素
92  int QueueRear(QUEUE *q, void *e)
93  {
94      if (QueueEmpty(q))
95      {
96          return -1;
97      }
98      QUEUENODE *node = q->head.prev;
99      memcpy(e, node->key, q->keySize);
100     return 0;
101 }
```