```
#include <stdlib.h>
#include <stdbool.h>

// queue implemented with a doubly linked list
typedef struct qNode{
    int val; // for row and col, use nodeID
             // nodeID = (row*maxCol) + col;
    struct qNode* prev;
    struct qNode* next;
}qNode;

typedef struct queue{
    int size;
    qNode* head; // remove from here
    qNode* tail; // insert here
}queue;

queue* createQ()
{
    queue* newQ = (queue*)malloc(sizeof(queue));
    newQ->size = 0;
    newQ->head = NULL;
    newQ->tail = NULL;
    return newQ;
}

void pushQ(queue* q, int val)
{
    qNode* newNode = (qNode*)malloc(sizeof(qNode));
    newNode->val = val;
    newNode->prev = NULL;
    newNode->next = NULL;

    if(q->tail == NULL)
    {
        // q is empty
        q->tail = newNode;
        q->head = newNode;
    }
    else
    {
        // q not empty
        newNode->prev = q->tail;
        q->tail->next = newNode;
        q->tail = newNode;
    }
    q->size++;
}

void popQ(queue* q)
```

```
{
    if(q->head != NULL)
    {
        // q is not empty
        qNode* temp = q->head;
        if(q->head == q->tail)
        {
            // q has 1 element
            q->head = NULL;
            q->tail = NULL;
        }
        else
        {
            q->head = q->head->next;
            q->head->prev = NULL;
        }
        q->size--;
        free(temp);
    }
}

qNode* topQ(queue *q)
{
    return q->head;
}

bool qIsEmpty(queue *q)
{
    return (q->head == NULL);
}

void freeQ(queue* q)
{
    qNode* temp;
    while(q->head != NULL)
    {
        temp = q->head;
        q->head = q->head->next;
        free(temp);
    }
    free(q);
}
```