```
/* Implementation of a simple circular queue using a static array */
#include <stdio.h>
#include <stdlib.h>
#include "queue.h"

/**
 * @brief Create the queue data structure and initialize it
 *
 * @param size
 * @return queue*
 */
queue *queue_init(int n)
{
        queue *q = (queue *)malloc(sizeof(queue));
        q->size = n;
        q->buffer = malloc(sizeof(cmd_info) * n);
        q->start = 0;
        q->end = 0;
        q->count = 0;

        return q;
}

/**
 * @brief Insert an item into the queue, update the pointers and count, and
 * return the no. of items in the queue (-1 if queue is null or full)
 *
 * @param queue
 * @param item
 * @return int
 */
int queue_insert(queue *q, cmd_info *item)
{
        if ((q == NULL) || (q->count == q->size))
                return -1;

        q->buffer[q->end % q->size] = *item;
        q->end = (q->end + 1) % q->size;
        q->count++;

        return q->count;
}

/**
 * @brief Delete an item from the queue, update the pointers and count, and
 * return the item deleted (-1 if queue is null or empty)
 *
 * @param queue
 * @return int
 */
cmd_info *queue_delete(queue *q)
{
        if ((q == NULL) || (q->count == 0))
                return (cmd_info *)NULL;

        cmd_info *x = (cmd_info *)malloc(sizeof(cmd_info));
        *x = q->buffer[q->start];
        q->start = (q->start + 1) % q->size;
        q->count--;

        return x;
}

/**
 * @brief Display the contents of the queue data structure
 *
 * @param queue
 */
```

```
void queue_display(queue *q)
{
        int i;
        if (q != NULL && q->count != 0)
        {
                printf("queue has %d elements, start = %d, end = %d\n",
                           q->count, q->start, q->end);
                printf("queue contents: ");
                for (i = 0; i < q->count; i++)
                        printf("\"%d\" ", q->buffer[(q->start + i) % q->size].jobid);
                printf("\n");
        }
        else
                printf("queue empty, nothing to display\n");
}

/**
 * @brief Delete the queue data structure
 *
 * @param queue
 */
void queue_destroy(queue *q)
{
        free(q->buffer);
        free(q);
}
```