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

/* create the queue data structure and initialize it */
queue* queue_init(int n) {
        queue *q = (queue*) malloc(sizeof(queue));
        q->size = n;
        q->buffer = malloc(sizeof(job_t) * n);
        q->start = 0;
        q->end = 0;
        q->count = 0;

        return q;
}

/* 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) */
int queue_insert(queue* q, job_t* 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;
}

/* delete an item from the queue, update the pointers and count, and
   return the item deleted (NULL if queue is null or empty) */
job_t queue_delete(queue* q) {
        if ((q == NULL) || (q->count == 0)) {
            job_t null_struct;
            null_struct.name = NULL;
            null_struct.number = -1;
            null_struct.wait_or_run = -1;
            return null_struct;
        }

        job_t x = q->buffer[q->start];
        q->start = (q->start + 1) % q->size;
        q->count--;

        return x;
}

char* job_to_string(job_t job) {
    char buf[BUFSIZ];
    if (job.wait_or_run == 0) {
        sprintf(buf, "%d\t%s\t\t%s", job.number, job.name, "Waiting");
    }
    else if (job.wait_or_run == 1) {
        sprintf(buf, "%d\t%s\t\t%s", job.number, job.name, "Running");
    }
    char* buff = buf;
    return buff;
}

/* display the contents of the queue data structure */
void queue_display(queue* q) {
        int i;
        if (q != NULL && q->count != 0) {
                for (i = 0; i < q->count; i++)
                        printf("%s\n", job_to_string(q->buffer[(q->start + i) % q->size
]));
                printf("\n");
        }
        else {
        printf("Nothing running or waiting.\n");
        }
}

/* delete the queue data structure */
void queue_destroy(queue* q) {
        free(q->buffer);
        free(q);
}
```