3) (10 pts) DSN (Queues)

A queue is implemented as an array. The queue has the 2 attributes, ***front*** and ***size***. ***front*** is the index in the array where the next element to be removed from the queue can be found, if the queue is non-empty. (If the queue is empty, front may be any valid array index from 0 to 19.) ***size*** is the total number of elements currently in the queue. For efficient use of resources, elements can be added to the queue not just at the end of the array but also in the indices at the beginning of the array before front. Such a queue is called a circular queue. A circular queue has the following structure:

```
typedef struct {
    int values[20];
    int front, size;
} cQueue;
```

Write an enqueue function for this queue. If the queue is already full, return 0 and take no other action. If the queue isn't full, enqueue the integer `item` into the queue, make the necessary adjustments, and return 1. Since the array size is hard-coded to be 20 in the struct above, you may use this value in your code and assume that is the size of the array values inside the struct.

```
int enqueue(cQueue* thisQ, int item) {

    if (thisQ->size == 20)          // 2 pts
        return 0;                   // 1 pt

    // 5 pts: 1 values, 3 pts index, 1 pt item
    thisQ->values[(thisQ->front+thisQ->size)%20] = item;

    thisQ->size++;                  // 1 pt
    return 1;                       // 1 pt
}
```

Grading notes: If they forget thisQ-> each time, take off 2 pts total.