Queues Programming Exercise

1. Provide an implementation of the Queue ADT in C using a doubly linked list, with nodes storing integer data. The ADT should support the following methods:

0. **Queue* create_Queue() ; /* creates an empty queue that you can push values into. Returns pointer to queue. */**

a. void enqueue(**int value**, Queue* Q)

b. int dequeue(Queue* Q) **/* returns value dequeued . You have to decide how to handle dequeue called on an empty queue.*/**

c. int top(Queue* Q)

d. int is_empty(Queue* Q) // returns 1 if empty, 0 if not empty

e. Queue* clone_queue(Queue* Q)

This operation will copy over an entire queue and return a pointer to the new queue.

f. Queue* sort_queue(Queue* Q)

This operation sorts the queue; with the minimum element at the front of the queue.

g. int play_queue(Queue* q, int k)

Consider the elements in the queue as a group of friends gathering to play hide-and-seek, and want to decide who is the person who will seek. To do this, they imagine themselves as being in a circle, and start counting from 1…k; 1 being the top of the queue. The k-th person from the start is removed. The count restarts from the next person, and goes around the circle to remove the second person in a similar way. This goes on, until only one person remains, who will be the seeker.

The above method returns the *position* of such a person in the queue (the top of the queue being 1). Your method should not destroy or modify the queue in any way. You may create a copy of the queue while doing this, if required.

h. Queue* remove_duplicates(Queue*)

Removes duplicates from the queue; i.e. removes integers that are repeated in the queue. If a number is repeated, you should keep the first occurrence of the number in the queue from the front. E.g. if the queue is [last] →4 5 4 5 5 20 3 3 4→[first], the queue without duplicates is:

→5 20 3 4→

This procedure modifies the queue in-place, and returns a Queue pointer to the modified queue.

Your implementation should have the following files:

1. queue-types.h : Defines the Queue* data type to store integer values.

2. queue-interface.h: Defines the interface based on the above functions

3. queue-list-methods.c: implements the functions defined in the interface.

Comment your code well, and include a text file explaining your solution to the problems briefly. Upload a single zip file named <roll-number>.zip as your submission.