CS212 – Lab 6 – Implementing a queue with pointers

A. Modify your array-based queue to use a pointer-based queue. All your original functions should remain in place, except they will now be using dynamically allocated nodes, as discussed in class. Define your anchor with the name "anchor", so we can quickly see it in your source. The functions below are the same as your original functions, but now must be modified to use a list. The queue can now grow to use up all available memory (you can have a huge number of elements in the queue). Your Node is a struct containing a "next" pointer as discussed in class, and a data item (a character).
   a. Push – adds one element to the end of the queue (accepts a character as input), returns an int:
      i. -1 if the queue is already full,
      ii. 0 if the item was successfully added
      iii. If the queue is empty (use "isFull" to find out) then create the first element of the queue
   b. Pull – removes the front element (if it exists) and returns the value:
      i. NULL if there were no elements (queue was empty)
      ii. The Front element itself, if there was one
   c. Front – does NOT remove anything. Returns the value:
      i. NULL if there are no elements (queue is empty)
      ii. The front char element if there is one
   d. isFull – does NOT remove anything. Returns an int:
      i. 0 if there are no elements (the queue is empty)
      ii. 1 if the queue is full
      iii. -1 for all other cases

B. Using the above functions, write a program that:
   a. Creates the anchor pointer for the stack of characters
   b. Uses your functions above to do the following:
      i. Accept input characters, 1 at a time until the symbol ^ is input
      ii. Push each character onto the queue
      iii. Output the queue (front to end) after all inputs have been received

Notes:

1. Your main program should have NO access to the queue except through the above functions. The queue & value pointer to the front and end ARE global, so all your functions can get at them, but the main code must never look at or use them.
2. Test the returned value from "malloc". If memory is full (the return is NULL) and more data keeps coming, your program should ignore the data and reply,

   "The queue is full. Enter the '^' character to stop."
3. Repeat the above message if more input (other than "^" keeps coming).