Using your existing **pointer-based** Stack & Queue functions (import them into a new file), write a new "main" function that evaluates an infix expression, using the algorithms in the slides on my website. The input is terminated by the "^" character. All incoming numbers will be single-digit integers. The input rules allow for at least 1, but possibly multiple spaces between numbers and operators. In the following steps, "print" means, display it on standard output. You may use 1-way or 2-way stacks & queues to fit your design.

1. Read a sequence of characters from standard input (could be console or a piped file or even output from another program) into a queue.
    a. Print the queue (so we know you got it all). When this is performed, the leftmost symbol printed will be the symbol that is at the "front" of the queue.
    b. Treat the first character inserted into the queue as the 1st character to be extracted in the next step.
2. Use the queue to "feed" the stacking operations to convert the sequence to postfix notation (algorithm 1).
    a. Print the entire stack (on one line) starting with the top of the stack after each "push" operation. When this is performed, the leftmost symbol printed will be the symbol that is at the "top" of the stack. Do not disturb the stack as you need it in the next step.
3. Print the final element that is on the top of the stack, as the answer to the evaluation. Label it as the answer (e.g.: "the answer is: " 42)

Rules:

Be sure to FREE list elements that are removed from your stack and queue

Operands are ONLY the usual: plus, minus, times and divide AND parentheses. Your evaluator should properly handle expressions like: ((2+3)*4)*(5+2)*(3+4)

Extra credit (10 points): DO NOT ATTEMPT THIS until you KNOW your regular operation is working. Allow for ** meaning exponentiation (2**3 means 2 cubed). This means you would have to be prepared to "peek" at any * to see if it is doubled to form the ** operator and be prepared to handle something like (2*3)**4 YOU will have to tell your CA to use the exponent test instead of the regular test.

hint: when you see the ** coming in, replace it on your stack with some other symbol that is NOT normally an arithmetic symbol (such as # or $) and treat it with higher priority than * or / in the evaluation algorithm.