Stacks and Queues

(1) Read through the following code. Draw the states of the variables `stack` and `queue` when indicated.

```
Stack<String> stack = new CarlStack<String>();
Queue<String> queue = new ArrayDeque<String>();
queue.add("tofu");
queue.add("broccoli");
queue.add("rice");
// (a) Draw the states of stack and queue
System.out.println(queue.element());
while (!queue.isEmpty())
{
   stack.push(queue.remove());
}
// (b) Draw the states of stack and queue
for (int i = 0; i < 2; i++)
{
   queue.add(stack.peek());
}
// (c) Draw the states of stack and queue
```

(2) We briefly talked about post-fix notation. Brainstorm with your partner how to *evaluate* a post-fix notation expression using a stack; that is, find the final value that the expression represents. Write your steps down in pseudocode (step-by-step English). Assume your algorithm will take a list as input. *[Hint: Read the next question.]*

(3) You receive the following list as input to your algorithm: `[3, 4, 2, /, 8, +, *]`. Walk through the steps of your algorithm with this input, and draw the state of the stack after each pop or push.

(over)

(4) **Palindromes** are words or phrases that read the same forward and backward, ignoring spaces, punctuations, or capitalisation (*e.g.*, “racecar” or “Sit on a potato pan, Otis!”). How would you use a stack and a queue to check whether a word or phrase is a palindrome? Write down an algorithm (in pseudocode) to check if something is a palindrome using these two data structures.

Extra time? Think about how to evaluate *infix* expressions, convert between infix and postfix, implement a queue using two stacks, implement a stack using two (or one!?) queues; or write Java code to implement your pseudocode algorithms.