Homework 2: Stack and Queue

Problem Description

1. Given an empty stack, what will be in the stack after the following operations?

- push(9)
- push(2)
- pop()
- push(pop()*7)
- push(30)
- push(pop()/3)

2. Given an empty stack, what will be the result after the following operations?

- push(1)
- push(pop()+3)
- push(pop()/2)
- pop()
- pop()

3. Given a stack s, S is s.push(s.pop() + s.pop()) and T is s.push(s.pop() * s.pop()). What will be the result after the following operations?

- s.push(1) -> s.push(2) -> s.push(3) -> s.push(4) -> s.push(5) -> S -> T -> S -> T

4. Suppose you can model a line ( ) as a stack. The top of the stack to represent the end of the line, and the bottom of the stack to represent the front of the line. If a new person comes, he/she will be the top of the stack. When you are ready for servicing someone, he/she will be removed from the bottom of the stack. If there are 200 people waiting in line, how many operations on the stack will be required in order to service the line's first person ?

5. Given a stack with 15 elements and a queue with 30 elements, performing the following set of operations 60 times:

- enqueue(pop()) /*remove the top element of the stack and insert it into the queue
- push(dequeue()) /*remove the last element of the queue and insert it into the stack

How many of the 45 total elements are involved in this process?

(Note: This is NOT asking how many times elements move, but how many elements move.)

6. Given two stacks (stack1, stack2), to implement a queue, you can assume stack1 will always contain the data in the correct order, i.e., stack1 will represent the tail of the queue where data is added and the bottom will represent the head. Removing from the top of stack1 will always give the most recently added data. In contrast, when stack2 contains data, it is reversed, with its top representing the head of a queue where data is removed. Removing from its top will always give the oldest data still stored. You can also assume that at any given time at least one stack is completely empty.

With above assumptions, you may use the following to implement the dequeue() function:

If stack1 contains data, then pop all the elements off of stack1 and put them on stack2. Pop an element off of stack2.

And, you may implement the enqueue(i) function as follows:

If stack2 contains data, then pop all the elements off of stack2 putting them on stack1. Push element onto stack1.

Questions: 1) Will the above algorithm work? If it does not work, please fix it and provide details; 2) Consider the above algorithm or your modified version, what is the time complexity of an enqueue operation directly after a dequeue operation?

Please submit the PDF version to assistant.

Solution: Your answers and proofs go here. Note that all math symbols should be in $$, such as $\div$, or use the "equation" command to generate a complex formula.