Checkpoint 0

Write a function to reverse a queue, using only the functions from the stack and queue interfaces.

```c
void reverse(queue_t Q) {
    // Hint: Allocate a temporary data structure
    while(__________________________) {
        ____________________________
        ____________________________
    }
    while(__________________________) {
    }
}
```

Checkpoint 1

Write a recursive function to count the size of a stack.

```c
int size(stack_t S) {
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    ____________________________
    }
```

Checkpoint 2

Why couldn’t this stack size implementation be used in contracts in C0?
Checkpoint 3

The above example works because function calls use a data structure that is like a stack. Step by step, trace out operationally the state of the computer's memory when it calculates the size of a stack with two strings “b” and “c”, taking account of the fact that each recursive call gets its own copy of the assignable variables.

Checkpoint 4

In the same fashion, trace out what happens operationally in this broken reversal function, starting with the code in main().

```c
void reverse(stack_t S) {
    string x;
    stack_t R = stack_new();
    while (!stack_empty(S)) {
        x = pop(S);
        push(R, x);
    }
    S = R;
}

int main() {
    stack_t S = stack_new();
    push(S, "foo");
    reverse(S);
    println(pop(S));
    return 0;
}
```