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(
    ) {
        // temporary data structure
    }
    while(
    ) {
        // temporary data structure
    }
}
```

Checkpoint 1

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

```c
int size(stack_t S) {
    // recursive implementation
}
```

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()`.

```
1 void reverse(stack_t S) {
  2   string x;
  3   stack_t R = stack_new();
  4
  5   while (!stack_empty(S)) {
  6     x = pop(S);
  7     push(R, x);
  8   }
  9
 10   S = R;
 11 }
 12
 13 int main() {
 14   stack_t S = stack_new();
 15   push(S, "foo");
 16   reverse(S);
 17   println(pop(S));
 18   return 0;
 19 }
```