15-122: Principles of Imperative Computation

R4: A queue_t Interface

Nivedita Chopra, Andrew Benson

Stack and Queue Interfaces

In lecture we discussed four functions exposed by the stack interface:

- stack_new: Creates and returns a new stack
- stack_empty: Given a stack, returns true if it is empty, else false
- push: Given a stack and a string, puts the string on the top of the stack
- pop: Given a stack, removes and returns the string on the top of the stack

Similarly, we discussed four functions exposed by the queue interface:

- queue_new: Creates and returns a new queue
- queue_empty: Given a queue, returns true if it is empty, else false
- enq: Given a queue and a string, puts the string at the end of the queue
- deq: Given a queue, removes and returns the string at the beginning of the queue

Checkpoint 0

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

```
1  void reverse(queue_t Q) {
2
3      _____________________________________________ // Hint : Allocate a
4                                                    // temporary data structure
5      while(_____________________________________) {
6
7          ___________________________________________
8
9          ___________________________________________
10     }
11     while(_____________________________________) {
12
13         ___________________________________________
14
15         ___________________________________________
16     }
17 }
```

Checkpoint 1

Write a recursive function to count the size of a stack. You may not destroy the stack in the process - the stack's elements (and order) must be the same before and after calling this function.

```
int size(stack_t S) {

    _____

    _____

    _____

    _____

    _____

    _____
}
```

Checkpoint 2

Why couldn't this stack size implementation be used in contracts in C0? Hint: Contracts in C0 cannot have side effects.

Checkpoint 3

The reversal function below is broken. Step by step, trace out the state of the computer's memory as this program executes, starting with the code in `main()`. Also, determine why the function does not work.

```
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;
}
```