The first page of your homework submission must be a cover sheet answering the following questions. Do not leave it until the last minute; it's fine to fill out the cover sheet before you have completely finished the assignment. Assignments submitted without a cover sheet, or with a cover sheet obviously dashed off without much thought at the last minute, will not be graded.

- How many hours would you estimate that you spent on this assignment?

- Explain (in one or two sentences) one thing you learned through doing this assignment.

- What is one thing you think you need to review or study more? What do you plan to do about it?

The first two questions on this problem set are due to Jeff Erickson: `http://www.cs.illinois.edu/~jeffe/teaching/algorithms`.

Question 1. Suppose we are maintaining a data structure under a series of n operations. Let $f(k)$ denote the actual running time of the kth operation. For each of the following functions f, determine the resulting amortized cost of a single operation. For amortized costs other than $\Theta(1)$, be sure to argue why your cost is also a *lower* bound, *i.e.* why it is not possible to do any better.
Hints:

- Start by either making tables of values and looking for patterns, and/or trying to write down a mathematical expression for the total cost $f(1) + f(2) + \cdots + f(n)$.
- If you notice a pattern, you could try proving it using the accounting method.
- Alternatively, if you are able to write down a mathematical expression for the total cost and simplify it, divide by n to get the amortized cost of a single operation.

1(a)

(a) $f(k)$ is the largest integer i such that k is evenly divisible by 2^i.

(b) $f(k) = k$ if k is a power of 2, and $f(k) = 1$ otherwise.

(c) $f(k) = k$ if k is a Fibonacci number, and $f(k) = 1$ otherwise.

(d) $f(k) = k$ if k is a perfect square, and $f(k) = 1$ otherwise.

Question 2. A *doubly extensible array* is a data structure that stores a sequence of items and supports the following operations:

- ADDTOFRONT(x) adds x to the *beginning* of the sequence.
- ADDTOEND(x) adds x to the *end* of the sequence.
- LOOKUP(k) returns the kth item in the sequence (0-indexed), or NULL if the current length of the sequence is less than or equal to k.

2

Describe and analyze a *simple* data structure that implements a doubly extensible array. Your ADDTOFRONT and ADDTOBACK algorithms should take $O(1)$ *amortized* time, and your LOOKUP algorithm should take $O(1)$ *worst-case* time. The data structure should use $O(n)$ space, where n is the current length of the sequence.

Question 3. Describe how to implement a queue using two stacks and $O(1)$ additional memory, so that the amortized time for any enqueue or dequeue operation is $O(1)$. The *only* access you have to the stacks is through the standard methods PUSH, POP, and ISEMPTY; in particular you should not assume anything about how the stacks are implemented. You may assume that PUSH and POP take $O(1)$ time in the worst case.