CS 181b
Advanced Topics in Algorithms
Spring 2002
Problem Set 3b
Due Tuesday, February 12 in class

1. **[20 Points] LFD is an Optimal Offline Paging Algorithm.** In class we mentioned that LFD (Longest-Forward-Distance) is an optimal offline algorithm for the paging problem. Recall that the algorithm works as follows: On each page fault, the algorithm evicts the page from fast memory whose next request is latest.

   It may seem intuitive that LFD is optimal, but intuition on these problems is often misleading! Here we'll prove the optimality of LFD formally in two parts.

   (a) First, consider the following claim:

   Claim 1 *Let ALG be any offline paging algorithm. Without loss of generality (as we saw in class!), ALG is a demand paging algorithm. Let σ be any request sequence. For any i, $1 \leq i \leq |\sigma|$, it is possible to construct another offline algorithm ALG_i that satisfies the following properties:*

   i. ALG_i processes the first $i-1$ requests exactly as does ALG.

   ii. If the i^{th} request in σ is a page fault, then ALG_i evicts from memory the page with the longest forward distance.

   iii. $ALG_i(\sigma) \leq ALG(\sigma)$.

   Prove this claim. (*Note:* To do so, you will want to specify how ALG_i behaves with respect to ALG after the i^{th} request.) Make sure that your proof is entirely rigorous.

   (b) Now show how the above claim can be applied repeatedly to prove that LFD is optimal.

2. **[15 Points] LFU and LIFO are not Competitive!** Prove that LFU and LIFO are not competitive online paging algorithms. LFU, Least-Frequently-Used, is a demand paging algorithm which evicts the page which has been used least since entering fast memory. LIFO, Last-In-First-Out, evicts the page that was most recently moved into fast memory. To show that these algorithms are not competitive, you must show that there is no constant c for which these algorithm are c-competitive.

3. **[15 Points] Conservative Paging Algorithms.** An online paging algorithm using a fast memory of size k is said to be **conservative** if for any request sequence, every subsequence containing k or fewer distinct page references incurs at most k page faults. (A subsequence is a consecutive sequence of requests.)

   Note that the definition of a conservative algorithm is similar, but not the same, as the definition of a marking algorithm. Prove that every conservative online algorithm is k-competitive.

4. [**20 Points**] **Adventures at Millisoft!** You have been hired by Millisoft as Director of Advanced Algorithms Design. You answer directly to the big guy, Gill Bates. Your first job is to help with the algorithmic issues on Millisoft's new operating system, Portholes 1998, which will be released in the very near future. On the first day on the job, Gill comes to your office sipping on a Diet Coke and sits in your bean bag chair. "I'd like to add a feature to Portholes 1998," he says, taking a chug of Diet Coke. "The user should be able to choose the paging algorithm that they want to use when they first install Portholes. Before we do this, we need to know whether or not each paging algorithm is a marking algorithm and whether or not it is a conservative algorithm." Fill in Gill's table below, putting a "Yes" or "No" in each cell. For any cell that we "filled in" in class, you may just put the answer down followed by a $\star$ to indicate that this was already shown. For each remaining cell, fill in the answer and then give a careful and rigorous proof. If an algorithm is not a marking algorithm or is not conservative, you should demonstrate how it can violate those definitions.

Algorithm	Marking	Conservative
LRU		
FIFO		
CLOCK		
FWF		