CS 182
Advanced Topics in Algorithms
Spring 2003
Problem Set 4a
Due Thursday, February 13 in class

1. **[15 Points] The Ski Rental Problem.** Professor I. Lai has a friend, Professor Sue Perfast, who is visiting him for the winter. Professor Perfast is a ski fanatic. Being the accommodating host that he is, Professor Lai agrees to go skiing with Professor Perfast whenever she wishes.

   Since he doesn't know in advance how many times he will be asked to go skiing, Professor Lai's dilemma is this: How many times should he rent skis before taking the plunge (so to speak) and buy a pair of skis? You may assume that it costs $30 to rent a pair of skis for a day and $300 to buy a pair of skis. *Moreover, you cannot buy the skis on the same day that you plan to use them – there isn't time for that! So, if you decide to purchase a pair of skis, there is a chance that you'll never use them again.*

   Professor Lai needs an online algorithm! The algorithm gets a "request to go skiing". If Professor Lai has not already purchased a pair of skis on an earlier day, then he must rent. If he bought skis earlier, he will use them. Such an algorithm would rent for the first j times (for some j) and would then buy skis on the evening after the j^{th} ski trip. The objective, of course, is to minimize the total cost incurred. An offline algorithm would know in advance how many "requests to go skiing" there will be and could therefore decide to either rent all the time or buy before the first ski trip.

   (a) Show that there cannot exist an online algorithm for this problem that has a competitive ratio better than 2. In other words, show that for any online algorithm for this problem, there exists some request sequence that causes the algorithm to pay at least twice the optimal amount.

   (b) Give an online algorithm for this problem which is strictly 2-competitive. Assume that skis cost $30 to rent skis for a day and $300 to buy.

2. **[15 Points] The Dynamic List Access Problem.** In class we looked at the MTF online algorithm for the List Access Problem. Specifically, we looked at the **Static** List Access Problem where only FINDs were allowed - there were no INSERT and DELETE operations permitted in the request sequence.

   Now, consider the more general **Dynamic** List Access Problem which differs from the static version in three ways:

   (a) Any FIND operation may fail. That is, the item might not be found. In this case, the actual cost is $\ell + 1$ where ℓ was the length of the list at the time that the FIND was performed.

   (b) INSERT operations are permitted in the request sequence. An inserted object is placed at the end of the list. If the length of the list prior to the INSERT was ℓ,

then the actual cost of this operation is $\ell + 1$. After inserting the element at cost $\ell + 1$, you may move it to any position earlier in the list at no cost (just like in a successful FIND operation).

(c) DELETE operations are permitted in the request sequence. The cost of the DELETE is simply the cost of finding the object. (However, the potential function changes as a consequence of the DELETE, so a DELETE differs from a FIND in this way!)

Show that when MTF is applied to the Dynamic List Access Problem, it still maintains a competitive ratio of 2. (That is, it is 2-competitive with respect to an optimal offline algorithm.)