Dear Robert,

Return by 3 P.M. Thursday, October 7.

I have included a diskette with my unique.exe and moveto.exe programs in subdirectory homework. The subdirectory data has three versions of the constitution, very short, short and full. Run times on DOS are as follows:

File	Number		Runtime	
	words	unique	unique	moveto
constitu.vsm	2322	618	1:36	0:47
constitu.sml	4529	917	4:40	1:52
constitu.ful	7587	1142	10:40	3:21

My program requires that the input be named test.txt, and outputs to the console. The runs for which the above data was copied were captured by I/O redirection and can be found in the files named runI.X, where 1 is 1, 2 or 3, and X is unq or mvt.

An important part of the grade this time is the program's efficiency. Therefore:

> *run all this week's assignments on the files constitu.vsm, constitu.sml and constitu.ful and grade they students accordingly.*

A program which cannot run the smallest of these receives at best a 3. Run consitu.ful only for the programs which seem worthy of it. Most will crash on constitu.vsm, and many will already by too time consuming on constitu.sml.

We are to impress upon the student the relative importance of *algorithmic efficiency*. This means looking at the constant factor of the leading term (and only the leading term) of the runtime as a function of problem size. What affects the leading term is what occurs inside the *inner loop* of the program. Here, the inner loop is the iterative searching through the list. Therefore, move-to-front pays off despite the involved pointer-pastings. Despite also if a student frees and reallocates a memory cell during the move-to-front. All these things affect the constant of the linear term, but the unique program was in fact $O(N^2)$, so none of that matters in the face of this reducing this higher order term.

The typical program is polynomial time.

> *It should become the student's second-nature to immediately look at what contributes to the highest order term of this polynomial and reduce or eliminate that.*

This is the most important concept so let's put our initial energy there.

In Pascal, efficient code and good code contradict each other. We will allow inefficiency in at least the following two areas:

1. Procedure calls in Pascal are slow. But we will not ask the student to minimize them.

2. Value passing of structures are slow. But we will not contradict our efforts have the student minimize var and global parameters into subroutines.

The extra credit is graded as 1 point for an acceptable program. Keep this point separate. As for the student's discussion, look for richness of the data they based their conclusions on and ignore entirely the conclusions themselves. We are not yet interested in the conclusions, but in the methods and fairness of the evalution.