We work on developing a selection of sorts.

1. Questions?

2. Sorting techniques we consider (see Lecture 22 for details):
   (a) Bubble sort. An attempt to reverse each out-of-order pair, in several passes.
   (b) Selection sort. Like bubble sort, but in each pass only the maximum value is exchanged to place it in its final position.
   (c) Insertion sort. A collection of sorted values is built up by inserting a new, random value in each pass. Compare with selection sort by observing the movement of maximum values.
   (d) Quicksort. A low and high values are segregated and then considered recursively.
   (e) Mergesort. A recursive sort that builds order from individual values upward.

3. A few notes on big-O notation.
   (a) We can frequently bound above some performance statistic by a mathematical function of problem size. Computer scientists are not concerned about the exact performance, but the dominant trend suggested by a curve.
   (b) When a function is a sum of several components, we ignore all by that component that is dominant in the limit. For polynomials, we select the leading term. Thus, a program that takes $n^2 + \log n$ time is described as an $O(n^2)$ algorithm.
   (c) Additionally, we generally don't concern ourselves with multiplicative constants. They can generally be ignored. Thus a program that makes use of $5n$ storage locations is described as $O(n)$ ("linear") in its space use.
   (d) Many of our early programs ("Hello world", "counting to 10") have used constant amounts of space. We write a constant bound as $O(1)$.
   (e) Programs that print out tables of simple calculations (like "counting to 10") frequently take $O(n)$ or "linear" amounts of time.
   (f) Tractable problems using one computer are typically limited to running time that is a small power of the problem size. For example, multiplying two $n \times n$ matrices seems to take no more than $O(n^{2.5})$ time, and requires no more than $O(n^2)$ space.
   (g) When we print the first $n$ integers, their width, the number of digits, grows as $O(\log n)$.
   (h) The total number of digits printed is $O(n \log n)$. Most fast sorts of $n$ values take $O(n \log n)$ time when using one computer.
   (i) We can pack $n$ equal-sized discs into a small square whose side is $O(\sqrt{n})$.  

*