1. **Number and title of course:** EECS 61B, Data Structures

2. **Course objectives:** In this course, you will study advanced programming techniques including data structures, abstract data types, interfaces, and algorithms for sorting and searching, and you will get a taste of “software engineering”– the design and implementation of large programs.

3. **Topics covered:**
   - Object-based and object-oriented programming
   - Java syntax and semantics
   - Designing classes
   - Algorithm analysis and asymptotic analysis
   - Testing, including unit testing
   - Use of a modern IDE for compilation control, program editing, and debugging (e.g. Eclipse).
   - The Java collections library as an example of class design
   - Sequences: arrays, linked lists, stacks, queues
   - Trees and traversals
   - Binary search trees
   - Priority queues
   - Hash tables
   - Balanced search trees, including at least B-trees (or red-black or 2-4 trees)
   - Pseudo-random number generators
   - Probabilistically balanced structures: skip lists
   - Sorting and selection
   - Graphs
   - Basic graph algorithms: depth-first traversal, topological sorting, shortest paths, minimal spanning trees
   - Union-find structures
   - Memory management

4. **Relationship of course to program objectives:**
   a. The programming-laboratory component of the course requires that students apply lecture material and readings either to small exercises or to substantial programming projects.
   b. The process of debugging involves forming hypotheses about where a particular malfunction occurs and then either probing execution states or examining sources to confirm the hypothesis. The process of testing is also a form of experimentation in which the hypothesis to be tested is that a program meets its specifications.
   c. The projects especially require that students design data structures and procedures to meet given sets of specifications, as well as to use components defined entirely by their interfaces.
   d. Not an objective addressed by this course.
   e. In implementing the projects, students must decompose the assignment into appropriate subsystems and implement each.
   f. Not an objective addressed by this course.
   g. As part of each project, students supply documentation that describes internals and externals of their programs.
h. Not an objective addressed by this course.

i. During this course, students can see that they use only a tiny portion of the features of the Java library and of the programming environment—that there is an enormous and growing amount of information they can never get in lecture and must actively pick up on their own.

j. Not an objective addressed by this course.

k. Students are exposed to basic program-development tools for compilation control, unit testing, debugging, and program construction, most recently as exemplified by the Eclipse IDE. The course introduces modern object-oriented design.

5. **Prepared by:** Paul Hilfinger (3/16/06)