1. **Department, number and title of course:** Electrical Engineering and Computer Sciences: CS 47B – Completion of Work in Computer Science 61B, Supplemental Data Structures

2. **Catalog Description:** (1 unit, self-paced, graded) Iterators. Hashing, applied to strings and multi-dimensional structures. Heaps. Storage management. Design and implementation of a program containing hundreds of lines of code. Students with sufficient partial credit in CS 61B may, with the consent of the instructor, complete the credit in this course.

3. **Prerequisites:** A course in data structures, plus CS 9B or equivalent, in addition to consent of the instructor. Students will not receive credit for CS 61B taken after CS 47B, or for CS 47B taken after CS 61B.

4. **Textbooks and/or other required material:**

5. **Course objectives:** Students who take CS 47B are expected to have had a course that familiarized them with the following: Arrays and linked structures, in particular with the efficiency tradeoffs between them; a variety of sorting algorithms for arrays and linked lists, including some $O(n \log n)$ sorts; binary search trees; and stacks and queues.

6. **Topics Covered:**
   - Course activities include programming assignments and quizzes; quizzes focus on low-level language details or programming techniques, while programming assignments are broader in scope. One of the programs is a substantial project comprising several hundred lines of code. The list of programming assignments appears below. All assignments use the Java programming language.
     - **Iterator exercises:** Students write a breadth-first and a depth-first iterator for a general tree.
     - **Hashing exercises:** Students analyze and devise hash functions applied to particular sets of data.
     - **Blocks project:** This project involves solving a sliding-block puzzle. Input data are an initial configuration and a goal configuration; the solution program must find a sequence of block moves that lead from the initial configuration to the goal. The project has a number of interesting features: it’s an example of backtracking search in which a hash table is used to avoid cycling; not only execution time, but also memory use must be accounted for in data structure and algorithm design.
     - **Heap exercises:** Students work with code to manage a binary heap.
     - **Storage management exercises:** Students modify a simulated storage manager, changing its first-fit strategy to best-fit and evaluating the result.

7. **Class/laboratory schedule:** Self-paced.

8. **Contribution of course meeting the professional component:** This course takes students who have learned the bare essentials of programming in previous programming courses and begins to expose them to the challenges and techniques involved in addressing larger programs. This exposure includes complexity analysis, object-oriented
design and abstraction techniques, use of modern IDEs, writing programs or components to detailed specifications, use of component libraries, and systematic testing.

9. **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.