This exam begins for you when you open (or peek inside) this packet. It ends at 11:59 PM on Thursday 2009 May 21. Between those two times, you may work on it as much as you like. I recommend that you get started early and work often. The exam is open-book and open-note, which means, precisely:

(1) You may freely consult all of this course's material: the Miller and Ranum textbook, your class notes, your old homework and exam, and the materials on the course web site. If you missed a class and need to copy someone else's notes, do so immediately, before either of you begins the exam. If you are not in possession of an old assignment and its grading comments (because your partner submitted it, for example) then obtain copies immediately, before either of you begins the exam.

(2) You may assume all concepts, algorithms, theorems, etc. discussed in class or in the assigned sections of the book. You do not have to define, develop, or prove them on this exam. On the other hand, you may not cite material that we have not studied. If you are unsure of whether you are allowed to cite material, just ask.

(3) You may not consult any other papers, books, microfiche, film, video, audio recordings, Internet sites, etc. You may use a computer for these four purposes: viewing the course web site materials, editing and running Python programs relevant to the course, typing up your answers, and e-mailing with me. You may not share any materials with any other student.

(4) **You may not discuss the exam in any way (spoken, written, pantomime, semaphore, etc.) with anyone but me until everyone has handed in the exam — even if *you* finish earlier.** During the exam you will inevitably see your classmates around campus. Please refrain from asking even seemingly innocuous questions such as "Have you started the exam yet?" If a statement or question conveys any information, then it is not allowed; if it conveys no information, then you have no reason to make it.

During the exam you may ask me clarifying questions. If you believe that the statement of a problem is wrong, then you should certainly ask for clarification. I will not be giving out hints.

Your written answers should be thorough, self-explanatory, and polished. Pictures and diagrams are often helpful. Always show enough work so that a classmate could follow your solutions. Do not show scratch work, false starts, circuitous reasoning, etc. If you cannot solve a problem, write a *brief* summary of the approaches you've tried. Submit your solutions in a single stapled packet, presented in the order they were assigned.

Partial credit is often awarded. When there are multiple solutions to a problem, it is understood that simple or efficient solutions may earn more credit than complicated or inefficient ones. Exam grades are loosely curved — by this I do not mean that there are predetermined numbers of As, Bs, Cs to be awarded, but rather that there are no predetermined scores required for grades A, B, C.

Good luck!

A. Give a concrete, detailed example of how a deletion from an AVL tree may require multiple rotations to restore the tree to balance.

B. The file `dictionarybst.py` implements `Dictionary` using a `BinarySearchTree`. This dictionary does not handle all of the key types (`int`, `float`, and `str`) that we want. Explain in detail how you would subclass this `Dictionary` to handle all of these keys. Your explanation may involve words, pictures, and/or Python code. Make it as precise as you can, so that I am certain that you understand all implementation issues. Also describe the efficiency of its `put()` operation (in terms of whatever quantities are relevant to its efficiency).

C. In a file `exam2.py`, rewrite the Solving Mazes assignment so that your program finds the shortest path from the starting point to the (any) ending point. As before, it should display the solution in dark blue and any other squares that were tried in light blue. If there are multiple shortest paths, then any of them is fine. Your program should work on any valid maze file. Of course it should be clear, well-written Python. Hand in your program electronically as usual.