NOTE: This exam has fewer questions, but more open-ended questions, than the previous exams in this course. I strongly recommend that you read through the entire exam before beginning and that you allocate time accordingly.
This exam is intended as a three hour sit-down examination. However, you may self-administer it at any time before 3pm on Wednesday May 10. Those who wish to take the exam during the scheduled examination period for this class may do so. Regardless of when you complete the exam, you may not discuss it with anyone until after all exams have been turned in or 3pm on Wednesday May 10 (whichever comes later). Exams should be turned in on paper to Holly Bennett in OC360 (or thereabouts) or by emailing them (via the scanner) to las.

The exam is intended to be completed in a single sitting. You may take more than three hours to complete the examination, but you should not consider this an unlimited-time exam. (Taking four hours would be fine, though presumably unnecessary; taking 20 hours would not. Exercise reasonable judgement.) In particular, anything you can’t solve within a reasonable amount of time is not likely to be worth an excessive effort.

This exam is closed book. **You are not permitted to use any materials, or to consult with any people, beyond the exam itself or the course instructor.** (I will be on campus Monday through Wednesday and should be available by phone, email, and IM for the duration of the exam interval. Please exercise reasonable discretion and don’t call outside of the hours of 8am-10:30pm.) You may also use a computer to type the exam, but should not use one for communication of any kind except with the instructor.

You may take reasonable breaks during the exam, but you are expected to honor the spirit of the single sitting administration. If possible, avoid mealtimes, conversations, phone calls, IMs, and other interpersonal interactions, though you may get up and walk around, have a snack, etc.

It is perfectly acceptable – even preferable – to hand write your answers in this exam booklet or on blank paper that you provide. Since you will be scanning the exam, you should be sure to write with a dark pen or heavily in pencil so that your writing is legible in the resulting pdf. If you wish to type your exam, you may use a computer but (a) you must not use resources on the computer other than your word processor (b) you should avoid checking email during the exam, except if that is your means of contacting me, and then only to read my email (c) you should not IM with people other than me.

Whether your exam is typed or hand written, each problem should be clearly identified, separated from other problems, and legible. Any extra pages should be stapled _in order_ to the back of this exam and, on the problem page in this booklet, you should write “see attached page (#).” You may also continue solutions on the backs of pages or on additional pages, but these should also be clearly labeled and the exam book should note where the solution can be found. (Scanned exams may omit staples.)

After you have finished this exam, in the space provided on the final page or on an attached piece of paper, please write out the phrase “I have neither given nor received unauthorized assistance during the completion of this work. I agree not to discuss this exam in any way until after 3pm on May 10.” Please sign your name to indicate that you have abided by all rules and conducted yourself according to the Olin College Honor Code. If you cannot write out this phrase and sign your name to it, please explain.¹

¹ This text courtesy of Professor Sarah Spence Adams.
1 Turing Machines

Assume that T1 and T2 are Turing machines that halt on any input, i.e., they always either accept or reject, never go into infinite loops. Call the language accepted by T1 L1 and the language accepted by T2 L2. Show how to construct a Turing machine for each of the following languages.

Your answers may be in English prose but should be sufficiently precise that it is clear how to actually construct the new Turing machine.
If you prefer, you may spell out the machine construction assuming that

\[ T1 = < Q1, \Sigma_1, \Gamma_1, \delta_1, q01, qaccept1, qreject1 > \]

and

\[ T2 = < Q2, \Sigma_2, \Gamma_2, \delta_2, q02, qaccept2, qreject2 > \]

A. \( L1 \cap L2 \), i.e., strings that would be accepted by both T1 and T2
B. $L_1 \cup L_2$, i.e. strings that would be accepted by either $T_1$ or $T_2$

C. $L_1$ complement, i.e., $\Sigma^* - L_1$
2  Applied Data Structures and Algorithms

For this problem, you should imagine that you are a member of a start-up company designing a new social networking application called YourPlace. YourPlace will provide each of its members with an individually controlled web space including blogging pages, collections of friends with pointers to their YourPlace pages, and other features common to the current trends in social networking/personal webspace. Since YourPlace isn’t fully designed yet, you are welcome to invent some features, but please explain anything that might not be obvious clearly enough that I can understand any new features that are important to your answers below.

A. Think about the data structures and algorithms that we’ve covered this semester. Identify three specific data structures or algorithms and illustrate each one by explaining how it could be used in the implementation of YourPlace. The explanation should name the data structure or algorithm, give a brief description of it, and show how it would be useful/what properties it would supply to YourPlace.

For example, if the question were “explain how each one could be used in a school’s registration system”, an answer might be:

Linked lists of student IDs could be used in the registration system to keep track of each class list. Students could be added to classes or removed from classes using linked list insertion and deletion. This would be a good choice because it would allow for easy modification of class lists, which happens a lot during registration.

If you choose a data structure or algorithm or part of YourPlace that’s more complicated, you may need to supply a slightly longer answer, but I would expect most answers to be well under half a page for each of the three illustrations. You may include pictures if they help to make your explanation easier.

Your answers will be judged on

- The appropriateness of your choice of data structure or algorithm for the use to which you put it. (It need not be the only choice – algorithm application is full of tradeoffs – but it should be a sensible one.)
- The accuracy of your description, i.e., that each data structure or algorithm actually has the properties you attribute to it.
• The diversity of the three algorithms and data structures you choose to describe. For example, an answer that suggest three applications of the same data structure or algorithm is not as strong as one that suggests three very different data structures or algorithms. This is your opportunity to show off the range of what you learned this semester as well as your creativity and your ability to understand a significant software design problem. (Note that you do not have to solve all of the algorithm design problems involved in implementing YourPlace, though!!)

• The (appropriate) succinctness of your answer. Long rambly answers whose key points are difficult to extract will not be well received. If you need to brainstorm first, go ahead, and you may attach those pages to the end of the exam, but please make your answer relatively crisp. See the sample answer (to a similar question) above.

DATA STRUCTURE/ALGORITHM 1 for YourPlace
B. Imagine now that you are building a YourPlace crawler, i.e., a program that will visit a particular individual’s YourPlace web space – say Pat’s Place – and identify the YourPlaces of Pat’s friends. Parents who are concerned about their children’s use of YourPlace have developed a list of suspicious characters on YourPlace, and they want to use this crawler to see whether their children might be close to one or more of these suspicious crawlers.

Using your knowledge of algorithmic techniques, program the YourPlace crawler to find out whether there are any suspicious characters within three “friend” links of a particular starting point. For example, if Pat lists Chris as a friend and Chris lists Terry and Terry lists a suspicious person, Pat is within three links of that suspicious person. You may assume that all friend links are reciprocal – if Pat lists Chris as a friend, Chris lists Pat – and that the crawler has access to the content of each YourPlace space through operations such as Pat.friends(), which returns a list of Pat’s friends.

Your answer should include both an English justification of the approach that you take as well as actual code or pseudo-code for the crawler, paying particular attention to the order in which it does things and any structures that you need/assume. Your answer will be judged on the accuracy and appropriateness of the techniques that you suggest, your use of appropriate terminology, as well as in the thoroughness and correctness of your pseudocode.

A complete answer to this problem should be between half a page and a page.
3 Language Comparisons

Consider the following code excerpts:

```prolog
append( [X | L1 ], L2, [X | L3 ] ) :- append( L1, L2, L3 ).
append( [], L, L ).
```

and

```scheme
(define (append l1 l2)
  (if (null? l1)
      l2
      (cons (car l1) (append (cdr l1) l2))))
```

These two excerpts implement roughly the same functionality in two different languages.

Using these excerpts as examples, your job is to compare and contrast the implementation of append in these two programming languages. You may want to note the following:

- How the data structures involved are represented/manipulated.
- How/whether the process of appending is described.
- What the machine running these programs actually does.
- The relationships of inputs and outputs.
- How the result is “returned”, i.e., what other code that made use of this result would look like. (Imagine that the appended list would next be filtered for duplicates, e.g., or augmented with mailing addresses.... What would this code look like?
- What aspects of the problem each highlights.

You may also find it useful to make reference to a more procedural language approach such as the following, although you are not explicitly comparing this one to the others:

```prolog
append(l1 l2):
    while l1.next is not null
    l1 := l1.next
    l1.next := l2
```

Your answer should take the form of bulleted lists of ways that the languages are similar/the same and ways in which they differ. (You may wish to use a two or three column format: Prolog only, both, Scheme only; or same/different.) Your
answer will be judged on its accuracy as well as on its insight: Have you captured the main points of the two languages? Do you demonstrate good understanding of programming language design and implementation? A well organized answer will also make it easier for me to read and to grade, so you may wish to develop a draft first and to turn in notes along with a revised answer for grading. Your answer to this question should fit on a single page; I will grade longer answers that are clear and comprehensible, but only up to two pages of reasonably sized text.
This is the end of the required portion of the exam. However, you MUST write out the honor code declaration below. Also, please make certain that you have put your name on every page of this exam booklet and any attached pages.

**Honor Code Declaration**

Please write out and sign the honor code declaration from the instructions on page 2 of this exam in the space below or provide an explanation here as to why you cannot do so.