CCM Programming Challenge: List Processing

This programming challenge asks that you join me in the creation of a little list processing library. The learning goal to to wrap your mind around list and recursive list processing. The technical goal to to establish a useful library of list processing functions with which you are intimate.

You should, in an way, think of this assignment as a **puzzler**! Actually, it consists of two kinds of puzzles:

1. The first kind of puzzle is a "What does it do?" puzzle. In this kind of puzzle, I provide you with code, and I ask you "What does it do?". There are two ways to go about answering this: (1) read the code, and (2) run the code. Since Prolog code tends to be opaque, you might have better luck, initially, at least, with the latter! Still, at some point, you should do your best to comprenend the code. I posed several puzzlers of this sort for you to do.
2. The second kind of puzzler is of the "Write code to do it" sort. I posed just a half dozen puzzlers of this sort for you to do.

Specific Tasks

1. Read and study the code in the given file, the `lp.pro`, which is a redacted version of the file that you want to complete. A great way to study this file will be to carefully, mindfully type it in to a file of your own (see next task), and do your best to more fully document each predicate as you do. In addition to helping you to wrap your mind around what each predicate does, this will take you a long way towards generating the knowledge needed to write the required user's manual for this list processing library.
2. Carefully, mindfully enter the given, partially redacted text into a file called `lp.pro` in your own Prolog world. I tend to put libraries like this one in a directory called `libraries` in a directory immediately subordinate to the one in which I do my Prolog programming.
3. Start a Prolog process. Load the file. It should load. If it doesn't, fix it. Play around with it a bit by running each of the predicates. Fix any programs that should arise. Do your best to understand the behavior of each predicate.
4. Define the redacted Prolog programs: `min`, `max`, `sort_inc`, `sort_dec`, `alist` and `assoc` – being sure to give them a good going over as you do. **Constraint**: The sorts must be straight selection sorts which make good use of the `min` and `max` and `remove` predicates.
5. Craft a demo that applies each predicate a time or two or three (as appropriate), being sure that the applications correspond to the order of the definitions in the given file. (Thus, `write_list` will come first, `flatten` will come last, and the rest will be applied in the order that is consistent with this specification.)
6. Create a **user's manual and tutorial** for this collection of list processing programs, complete with title, introductory paragraph, table of contents (you needn't have page numbers), and an entry for each program containing notes on syntax and semantics and an in context demo of the program. (Please generate the in context demo within a Prolog process and simply copy and paste the demo to the users' manual document that you are asked to write.

Post

Please post the following items to your web work site in an "entry" for this assignment:

1. The code (complete with the six programs that you were asked to define)
2. The demo that applies each predicate some number of times in the specified order

3. The user's manual and tutorial

Due Date

Friday, October 9, 2020