CSE-6490B
Final Exam

In your submitted work for this final exam, please include and sign the following statement:

> I understand that this final take-home examination is supposed to be strictly individual effort. I certify, under the penalty of academic dishonesty, that I have not consulted with anyone other than the instructor in preparing my answers.

The exam is open-notes. It is out of a total of 50 points. Choose and do five of the following eight questions.

---

1. **General.** *Block that metaphor!* [analogy] (10 points)

   For each of the following, come up with a good answer for the *analogy*. For example, consider

   relational database : SQL :: XML document : _____

   This is to be read as "relational database *is to* SQL *as* XML *is to* _____." A good answer here is "XQuery". Why? A relational database is a collection of data that can be queried with the query language SQL. An XML document is a collection of data that can be queried with the query language XQuery.

   In each case, you may write a brief—one or two sentence—explanation behind your choice.

   a. (2 points) except : SQL :: _____ : Datalog¬
   b. (2 points) GAV : mediator :: LAV : _____
   c. (2 points) SQL : structured data :: _____ : semi-structured data
   d. (2 points) XQuery : XPath :: SQL : _____
   e. (2 points) SQL : RDBMS :: _____ : Semantic Web

2. (10 points) **Expressiveness.** *Express yourself!*

   a. (2 points) One could, albeit with much effort, code up chess via the *win* and recursion-through-negation like we did for the *stones* game in class.

   If our chess program is locally stratified, then this means that there is a perfect model, and everything is assigned *true* or *false*. This means *win* (*"beginning board state"*)is either *true* or *false*. So it would be known that white (the first player) could always win playing a perfect game *or* that black (the second player) could always win playing a perfect game.

   Does this mean that a game of chess is necessarily winnable by the perfect white player or the perfect black player? Why or why not?

   b. (2 points) Are there any types of queries that can be expressed in SQL but not Datalog?

   c. (2 points) Are there any types of queries that can be expressed in SQL but not Datalog¬? (Careful.)

   d. (2 points) Is Datalog a superset of first-order predicate calculus (logic)? Why or why not?

   e. (2 points) Is Datalog¬ interpreted under negation-as-finite-failure, the well founded semantics, or the stable model semantics a subset of first-order predicate calculus (logic)? Why or why not?

3. **Information integration.**[1] *Schema scheming.* [exercise] (10 points)

   We are trying to integrate a set of booksellers. The schema we want to export to the user involves just a single relation:

   Book(author, title, subject, price, #pages)

   Assume we have the following two online bookstore sources:

   A) *AliensRComing*: Only sells sci-fi books. For each book it sells, it can give the information about author, title, price, and the number of pages.

   E) *EndIsNear*: Primarily sells religious books. However, it also stocks a few books on other subjects. For each book it sells, it has information about author, title, subject, and price.

   a. (2 points) Consider integrating these sources using the global-as-view approach. Write down the view corresponding to the **Book** relation.

   b. (2 points) Consider the following SQL query:

   ```
   select author
       from Book
       where subject = "sci-fi"
   ```

   Show how it gets rewritten in terms of the sources in the global-as-view approach.

   c. (2 points) Now consider instead integrating these sources using the local-as-view approach. Show how the two sources will be modeled.

   d. (2 points) Consider your local-as-view approach from Question 3c and the following candidate plan for the query from Question 3b:

   > Compute the results by calling the source **A** and selecting the authors from the returned tuples.

   Show, using the idea of containment mappings, that this plan is in fact a sound plan for the query. Is it also a complete plan? If not, what additional information about the sources would make it a complete plan?

   e. (2 points) Consider your local-as-view approach from Question 3c. Show how the inverse-rule algorithm rewrites the query from Question 3b into a maximally contained plan.

[1] Thanks to Subbarao Kambhampati.

4. **Semantic Web.** *The web means so much to me.* [essay] (10 points)

   Compare and contrast XML and RDF Databases. What functionality does RDF provide, if any, that is different than what XML already provides? And, vice-versa, what functionality does XML provide, if any, that is different than what RDF provides?

   Keep your answer to under 500 words. (That is less than a page typed in ASCII.)

5. **Trees.** *Can't see the trees for the forest.* (5 points)

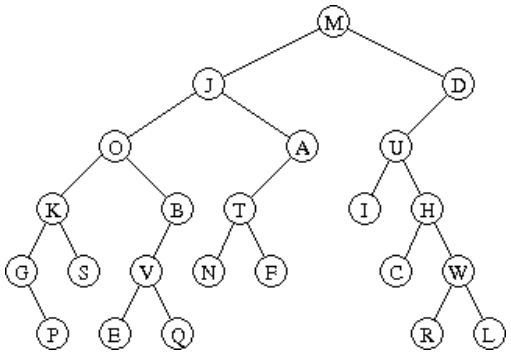
   Dr. Dogfurry, the infamous XML researcher, has the following idea to ship XML "trees" over the network. He suggests flattening the tree to a simple list. Then the tree can be reconstructed on the receiving side. Of course, simply a list of the nodes would not be sufficient to rebuild the tree. It is known that a *binary* tree can be uniquely recontructed given both its pre-order and in-order traversals. So Dr. Dogfurry suggests sending both the pre-order and in-order traversals as lists. (This is just for binary trees. The next step in the research for Dr. Dogfurry is to consider arbitrary trees.)

   On the receiving end, we need an algorithm to reconstruct the binary tree. Write in pseudo-code—in the language style of your choice—an algorithm to do the following. It takes as input two arrays, `Pre` and `In`. The first array represents the pre-order traversal of some binary tree. The second array represents the in-order traversal of the same binary tree. Its output should be a natural data-structure that represents the binary tree.

   For example, consider the following pre-order and in-order arrays:

   pre-order: `M J O K G P S B V E Q A T N F D U I H C W R L`
   in-order: `G P K S O E V Q B J N T F A M I U C H R W L D`

   The only binary tree that results in these pre-order and in-order traversals is



   Note that your algorithm is for binary trees in general, and not binary *search* trees. The example above resulted in a binary tree that is not a binary search tree. This had to be the case since the in-order traversal of the tree is not in alphabetical order of the labels.

   This is easiest to implement this via recursion.

6. **XPath.** *Oops, I think we're on the Y path by mistake.* (10 points)

   Compose the following XPath queries with respect to *Bibliography* XML document (on the course webpage with this assignment). Look over the Bibliography document to understand its "schema".

   Your queries ought to be logically correct in that they would still do the job as expected if the Bibliography document were to have many new papers added.

   a. Extract the titles of the paper.
   b. Return the titles of the papers by Halevy.
   c. Extract the authors of papers that appeared in a session named "Invited Talk".
   d. Extract the titles of papers that have four or more authors.
   e. Return the names of the conferences *and* journals.
   f. Extract the titles of papers such that "XML" is mentioned *somewhere* in the paper's ancestors's attributes or in the descendents's attributes or text.
   g. Return the titles of papers co-authored by Hass and Kossmann.
   h. Return the titles of papers that appeared after the year 2000.
   i. Return the titles of papers that are longer than 20 pages.
   j. Return the names of conferences that contain a paper with "XML" in its title and 'Chaberlin' as one of its authors.

7. **XQuery.** *So this is where the X path leads.* (10 points)

   Again, compose your queries with the *Bibliography* XML document in mind. I realize many will not have access to any XQuery engine—unlike, perhaps, XPath—and that you may not test these. So small syntax glitches and such will be ignored. However, you should get the logic of the query right.

   As before, your queries ought to be logically correct in that they would still do the job as expected if the Bibliography document were to have many new papers added.

   a. (2 point) Count the number of papers that appear each year in the Bibliography.

   b. (4 points) Return an HTML list of the authors who appear in the Bibliography. Under each author item, present a sub-list of that author's papers by title and year. E.g.,

      ⋮
      - Halevy, Alon
        - Schema mediation in peer data management systems, 2003.
          ⋮
      ⋮

   c. (4 points) In a list, report each author, and for each author, provide a sublist that enumerates his or her co-authors (other authors who have written a paper together with the author). E.g.,

      ⋮
      - Halevy, Alon
        - Ives, Zachary G.
        - Suciu, Dan
          ⋮
      ⋮

      Order the list and the sub-lists.

8. **XML & XPath.** *Extra marks longed for.* [analysis] (10 points)

   Consider an XML document of size N, so with a size on the order of $\mathcal{O}(N)$. That is, printing it in any reasonable way would take $\mathcal{O}(N)$ bytes.

   Assume each element in the document is size $\mathcal{O}(1)$, not counting the content and sub-trees of the element. Assume each element has $\mathcal{O}(1)$ attributes, each of size $\mathcal{O}(1)$.

   An XPath query on the document returns a *list* of the nodes extracted from the XML document that match the query.

   a. (4 points) What is the length of the list resulting from the XPath query '//*' applied to the document?

   b. (4 points) What is the *size* of the results in total from the XPath query '//*' applied to the document? Consider the worst case.

   c. (2 point) What is the *size* of the results in total from the XPath query '//*/@*' applied to the document? Consider the worst case.