CSCI 4230 – Assignment 7

Instructions

For this assignment you must write the following predicates in Prolog. For the first six problems you can assume that the predicates `parent`, `male`, and `female` have already been defined. I have provided a file called `assign7data.pl` containing these definitions.

1. Write a predicate `mother(Mother, Child)` that succeeds if `Mother` is the mother of `Child`.
2. Write a predicate `father(Father, Child)` that succeeds if `Father` is the father of `Child`.
3. Write a predicate `sibling(Sibling1, Sibling2)` that succeeds if `Sibling1` and `Sibling2` are siblings. Note that a person can not be their own sibling.
4. Write a predicate `first_cousin(Cousin1, Cousin2)` that succeeds if `Cousin1` and `Cousin2` are first cousins.
5. Write a predicate `ancestor(Ancestor, Descendant)` that succeeds if `Ancestor` is an ancestor of `Descendant`. You will need to define this one recursively.
6. Write a predicate `common_ancestor(Ancestor, Person1, Person2)` that succeeds if `Ancestor` is an ancestor of both `Person1` and `Person2`.
7. Write a predicate `do_reverse(List, Reverse)` that succeeds if `Reverse` contains the elements of `List` in reverse. You may not use the built-in predicate `reverse`.
8. Write a predicate `insert_item(Item, List, Result)` that succeeds if `Result` contains the elements of `List` with `Item` inserted and both `List` and `Result` are sorted. You do not have to verify that `List` is sorted, you may assume that it is instantiated with the value of a sorted list.
9. Write a predicate `insertion_sort(List, SortedList)` that uses `insert_item` to sort `List` by first recursively sorting its tail, then inserting the head into the result to get `SortedList`. You may not use the built-in `sort` predicate.
10. Write a predicate `is_union(Set1, Set2, Union)` that succeeds if `Union` is the union of `Set1` and `Set2`, where `Set1` and `Set2` are unsorted lists of unique items. You may not use the built-in `union` predicate, but you may use the built-in `member` predicate.
11. Write a predicate `is_intersection(Set1, Set2,Intersection)` that succeeds if `Intersection` is the intersection of `Set1` and `Set2`, where `Set1` and `Set2` are unsorted lists of unique items. You may not use the built-in `intersection` predicate, but you may use the built-in `member` predicate.

What to Hand In

Implement all of the functions described above in a source file called `yourlastnameAssign7.pl` with your actual last name. Make sure to put your name, CSCI 4230, and Assignment 7 in the comments (comments in Prolog start with `%` and go to the end of the line. Upload the source file to D2L to the dropbox called Assignment 7.