Overview

As is the case with many declarative/functional programming languages, Prolog provides an elegant and useful notation for dealing with structured data in the form of lists. At their most basic, lists are written as bracketed terms using a syntax similar to that of the Python language, for example: `[1,2,3]` denotes the list comprised of the terms `1`, `2`, and `3`; and the `[foo(x), bar(1), baz(a,b)]` denotes the list comprised of the terms `foo(x)`, `bar(1)`, and `baz(a,b)`.

Lists are defined recursively, and Prolog provides a specialized syntax for separating the head and tail of an arbitrary list using the syntax `[Head|Tail]`. This allows us to define various relations over lists such as length:

```prolog
% length/2
% length(L,N) - read as "N is the length of the list L"

% The length of an empty list is 0.
length([], 0).

% The length of a non-empty list is 1 + the length of its tail.
length([Head|Tail], N) :-
    length(Tail, M),
    N is M + 1.
```

Note that with regards to the second part of the definition, the actual head of the list is irrelevant. Because of this, we may rewrite the rule as follows:

```prolog
% The length of a non-empty list is 1 + the length of its tail.
length([_|Tail], N) :-
    length(Tail, M),
    N is M + 1.
```

Defining the membership relation between an element and a list is done in a similar manner[1]:

```prolog
% member/2
% member(X,XS) - read as "X is a member of the list XS".

% X is a member of the list whose first element is X
member(X, [X|_]).

% X is a member of the list [Y|Tail] iff X != Y and X is
% a member of the tail.

member(X, [Y|Tail]) :-
    dif(X, Y),
    member(X, Tail).
```

Lists are in some sense a *universal data structure*. The ability to nest them allows us to implement any conceivable data structure as a list of some sort. For this assignment, you will implement a set of relations over lists in Prolog

[1]Note that the second rule makes use of the built-in relation `dif/2`. The `dif/2` predicate is a constraint that is true if and only if `A` and `B` are different terms.

that allow us to talk about various operations on sets.

Assignment Specification

For this assignment you are to write a knowledge base in Prolog defining the following operations:

1. `set_union/3` - if X and Y are sets represented in list form, then $Z = X \cup Y$.
2. `set_intersection/3` - if X and Y sets represented in list form, then $Z = X \cap Y$
3. `set_difference/3` - if X and Y are sets represented in list form, then $Z = X \setminus Y$.
4. `is_subset/2` - if X and Y are sets represented in list form, then `is_subset(X,Y)` iff $X \subseteq Y$.

Grading Criteria

Overall, this programming assignment is worth a total of 100 points according to the following breakdown:

- 75 points will awarded on the basis of your program's ability to correctly answer queries regarding the family relationships discussed in lecture and specified in this assignment.
- 25 points will be awarded based on the clarity and elaboration tolerance of your program.