Lab 8

Due: July 21, 2023 at the end of lab

1 Lab setup

1.1 Importing project

The skeleton code at `https://classroom.github.com/a/6GZxNPMN` is the start of this week's lab. For the amount that we are using Prolog, it makes the most sense to download the source file manually and update it with your solution on Github manually.

1. In Github, go to your Lab 8 project
2. Click the **Repository** link in the horizontal navigation bar.
3. Click the **Download zip** button to download all the project files.
4. Extract the .zip file into an appropriate place on your computer.

At this point, you should have Lab 8 available to load into SWI-Prolog.

1.2 Completing project

Refer to section **2** for the programming part of the project.

1.3 Exporting project

Once you have completed your code (or you have updated it since last pushing it to Github), follow these steps to make sure that I can access your most recent code:

1. In Github, go to your Lab 8 project
2. Click the **Repository** link in the horizontal navigation bar.
3. Click on the file that you want to update.
4. Click on the **edit** button.
5. Paste your Prolog code into the text box.
6. Click the **Commit Changes** button at the bottom of the page.

Once you have committed your code, I have access to it and I can download and grade it. I look at the date of the most recent changes in github to determine timeliness of your lab, so make sure go through these last steps before the due date.

2 Lab 8 Specification

You should implement all of your facts and rules in **lab8.pl**. **DO NOT** use any builtin Prolog predicates for this lab.

2.1 List membership

Write a predicate named **isMember/2** that checks if the first argument is an element of the second argument. You should implement this similarly to our F# approach, which included a base case and a recursive case. In Prolog, however, we are only focused on provable knowledge. That means if we ask something like `?- isMember(x, [])`, it is false by default (as it should be) and we don't need to write a special case for it.

Use recursive list coding patterns and refer to the slides for syntax on lists.

Expected console output for some queries:

```
?- isMember(a, [a, b, c]).
true.

?- isMember(X, [a, b, c]).
X = a; X = b; X = c.

?- isMember(X, []).
false.
```

2.2 Set difference

Write a **setDiff/3** predicate that performs set subtraction, just as you did in F#. The three arguments should all be lists. The results (appearing in the third argument) should be a list of elements that appear in the first argument but not the second argument.

Since Prolog doesn't have return values, treat the third argument as the implicit return argument. Assume that all queries of **setDiff** will have a variable as the third argument, and that there are no duplicate values in the first list, and no duplicate values in the second.

Expected console output for some queries (you may also see a **false** result after the answers, and that is fine):

```
?- setDiff([a, b, c], [d, e, f], X).
X = [a, b, c].

?- setDiff([a, b, c], [b, c, d], X).
X = [a].

?- setDiff([a, b, c], [c, b, a], X).
X = [].
```

Hint: You will need separate rules for the recursive case, to cover when the head of the first list is in the second list, and when it is not.

3 Grading

Be professional. Make results easy to understand and grade. Include only those parts to be graded. Leave comments where necessary, especially if it aids in grading.

isMember and **setDiff** are each worth 50% of the lab grade.