MSCS-632: Assignment 7: Lab Report: Building a Family Tree Logic System in Prolog Using Inference and Recursion

----

Sandesh Pokharel

University of the Cumberlands

MSCS-632 Advanced Programming Languages

Dr. Vanessa Cooper

April 23, 2025

Introduction

This assignment involved designing a simple Family Tree program in Prolog that models and infers relationships between family members using logical rules and recursion. The goal was to define both basic and derived relationships (e.g., parent, grandparent, sibling, cousin), use Prolog's inference capabilities to answer relationship queries, and explore the power of recursion to find descendants.

GitHub Repository

All the source code is available in the public GitHub repository:

https://github.com/sanspokharel26677/MSCS-632-Assignment8

Implementation Process

I started by writing Prolog *facts* for each basic relationship — such as parent, male, and female. From there, I moved on to building *rules* using Prolog's logical syntax. These rules helped define grandparent, sibling, and cousin relationships. Finally, I implemented a recursive rule called descendant that allows Prolog to trace indirect relationships (e.g., grandchildren and great-grandchildren) by climbing down the family tree.

All logic is defined in the file family_tree.pl. The file was written with clean formatting and comments for clarity, following best practices for logical programming.

Challenges and Learning

This was my first time working with Prolog, so the syntax felt very different from object-oriented or procedural languages I'm used to. One challenge was understanding how recursion works in a declarative setting — especially with rules like descendant(X, Y) that require thinking "bottom-up."

Another challenge was debugging. Prolog doesn't throw standard error messages like Java or Python, so I had to learn how to interpret the response when queries failed due to missing facts or miswritten predicates.

Key Features Implemented

- Defined **parent**, **male**, and **female** relationships as base facts.
- Derived **grandparent** and **sibling** relationships through multi-level logic.
- Implemented **cousin** relationship using sibling checks on parents.
- Built a **recursive descendant rule** to model long-range ancestry.

All rules work using Prolog's built-in pattern matching and logical inference. I validated correctness by running a large set of test queries (see below).

Sample Queries

A variety of queries were tested to validate the logical relationships. These include:

- Who are the children of John?
- Is Alice a cousin of Elena?
- Who are the siblings of Mike?

- Is Elena a descendant of John?

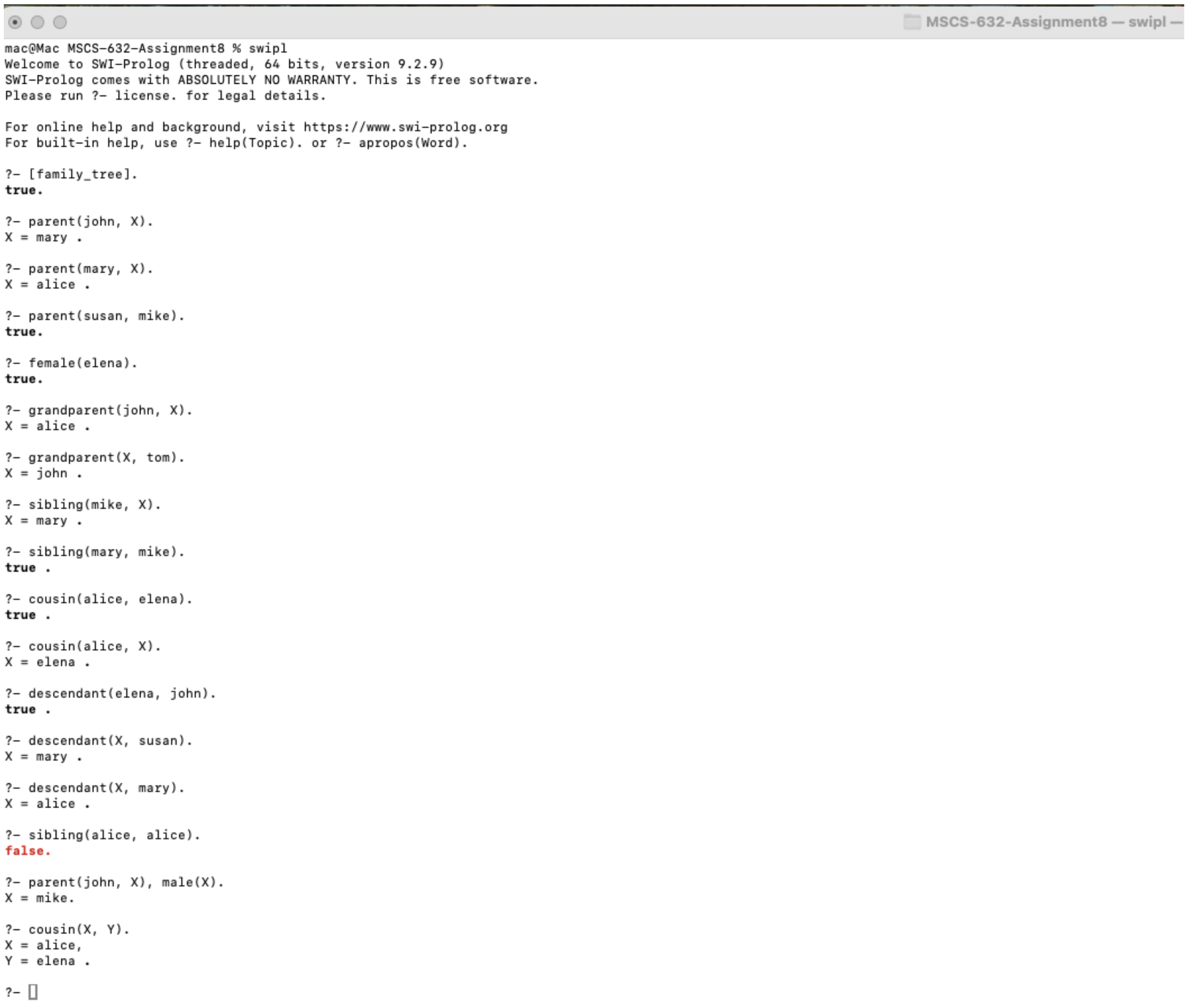
See the **Sample Queries and Expected Output** table in the appendix for a complete list.

Appendix: Sample Queries and Outputs

	Expected Output
parent(john, X).	X = mary ; X = mike.
parent(mary, X).	X = alice ; X = tom.
parent(susan, mike).	true.
female(elena).	true.
grandparent(john, X).	X = alice ; X = tom.
grandparent(X, tom).	X = john ; X = susan.
sibling(mary, mike).	true.
sibling(mike, X).	X = mary.
sibling(alice, alice).	false.
cousin(alice, elena).	true.
cousin(alice, X).	X = elena.

cousin(X, Y).	X = alice, Y = elena.
descendant(elena, john).	true.
descendant(X, susan).	X = mary ; X = mike ; X = alice ; X = tom ; X = elena.
parent(john, X), male(X).	X = mike.

Screenshots of Testing



```
mac@Mac MSCS-632-Assignment8 % swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [family_tree].
true.

?- parent(john, X).
X = mary .

?- parent(mary, X).
X = alice .

?- parent(susan, mike).
true.

?- female(elena).
true.

?- grandparent(john, X).
X = alice .

?- grandparent(X, tom).
X = john .

?- sibling(mike, X).
X = mary .

?- sibling(mary, mike).
true .

?- cousin(alice, elena).
true .

?- cousin(alice, X).
X = elena .

?- descendant(elena, john).
true .

?- descendant(X, susan).
X = mary .

?- descendant(X, mary).
X = alice .

?- sibling(alice, alice).
false.

?- parent(john, X), male(X).
X = mike.

?- cousin(X, Y).
X = alice,
Y = elena .

?- 
```

Conclusion

This assignment was a great introduction to Prolog and declarative programming. I was able to build a complete logical system from the ground up and understand how inference and recursion differ from imperative languages. Overall, this project helped strengthen my understanding of knowledge representation and rule-based logic systems.

References

Bratko, I. (2012). *Prolog programming for artificial intelligence* (4th ed.). Pearson Education.

Sterling, L., & Shapiro, E. (1994). *The Art of Prolog: Advanced Programming Techniques*. MIT Press.

SWI-Prolog. (n.d.). *SWI-Prolog official documentation*. https://www.swi-prolog.org