Project 4 – Logic proGramming with prolog

Programming Languages

# Objective

Build a mini expert system in **Prolog** for a student-selected application.

# Topics

* Logic Programming Languages
* Expert systems
  + Knowledge-bases + Inference engine
  + Applications

# Instructions

### Preliminary

Select a partner (recommended)

### Research expert systems and their applications.

See [references](#_References) at the end of this document to get started.

### application

1. **Select** an **application of interest** for which you’d like to build your own mini expert-system.
   1. To start brainstorming ideas **see what’s been done before in section on Applications in reference [1]** .
   2. Keep in mind that you will be **evaluated** based on your applications’
      1. **Originality** (don’t do what’s been done before; you can do something close, however, with a useful/meaningful twist)
      2. **Usefulness** (theoretical – don’t just pick a silly idea to satisfy the assignment)
      3. **Complexity**
2. **Review** your selection with **instructor** (this will be part of the grade).

Depending on the size of your selected application, your expert system may represent only a **subset of all possible facts and rules**.

1. Start to determine:
   1. The **facts and rules** that will make up the knowledge base of your expert system.
   2. The type and format of **queries** that will be posed against your expert system.

### prolog compiler

1. If you have not done so already, [download](http://www.swi-prolog.org/download/stable) and install [**SWI-Prolog**](http://www.swi-prolog.org/)
2. **Run SWI-Prolog**

### source file with facts and rules

1. In SWI-Prolog, select **“File/New”** to create a new Prolog source file that will contain the facts and rules for your knowledge base *(your source file will have the extension* ***.pl****).*
2. **Add rules** to your prolog source file
   1. Please see the simple [Family relationships example](#_Example) in order to see how rules/facts and queries work together in prolog.
3. **Add facts** to your prolog source file about specific objects.
4. **Compile** your source file

### test your expert system

1. Return to the main **SWI-Prolog window.**
2. Test out a large selection of goals, some of which return **true or false**, and others which return **value(s)** that make the goal true.

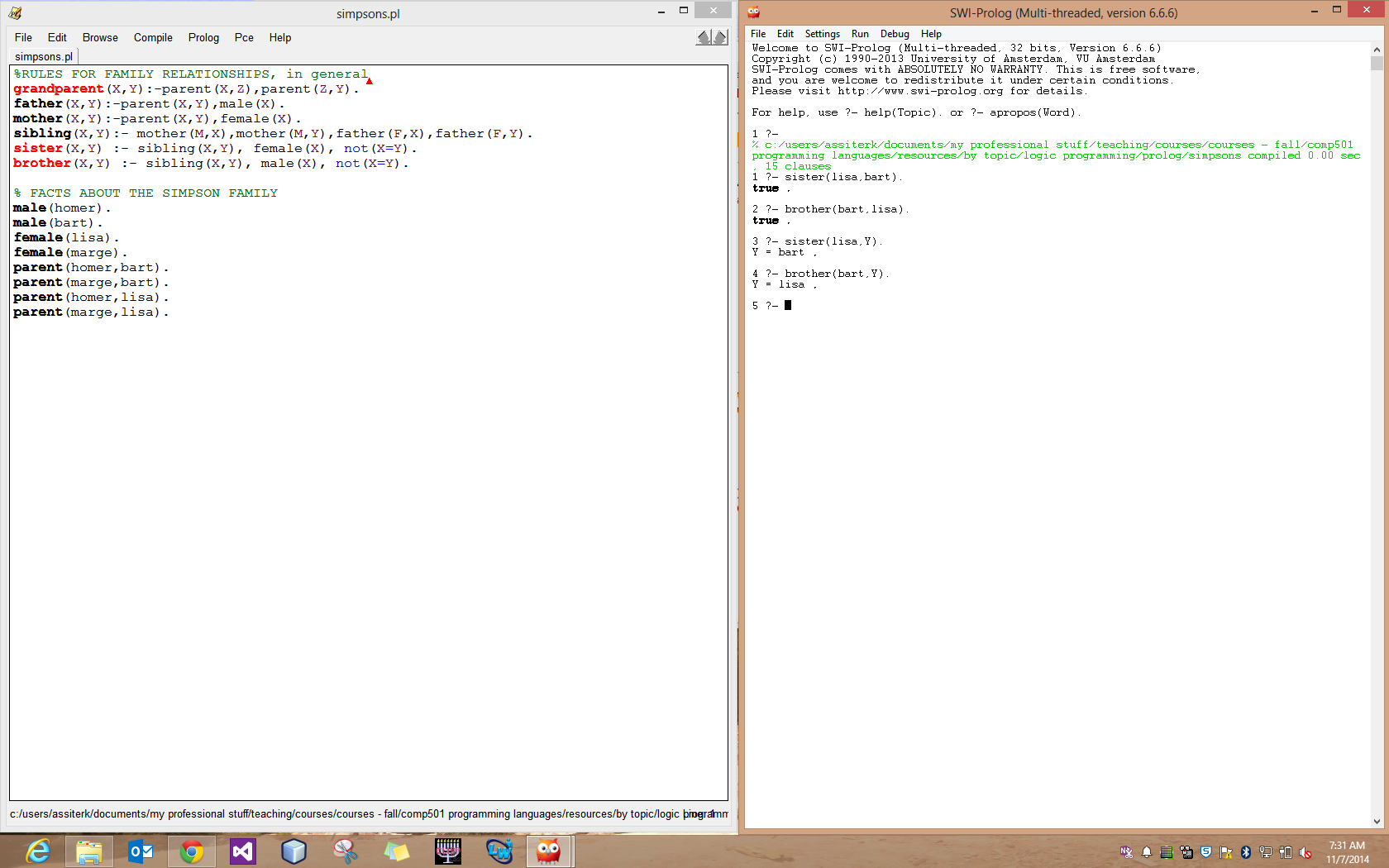
### Submit your work

1. Make sure that your name is in ***all***of your project files.
2. If you have **more than one file** for your solution, make a **.zip file** for your project
3. In **Blackboard**, attach your solution file to the submission for this assignment.

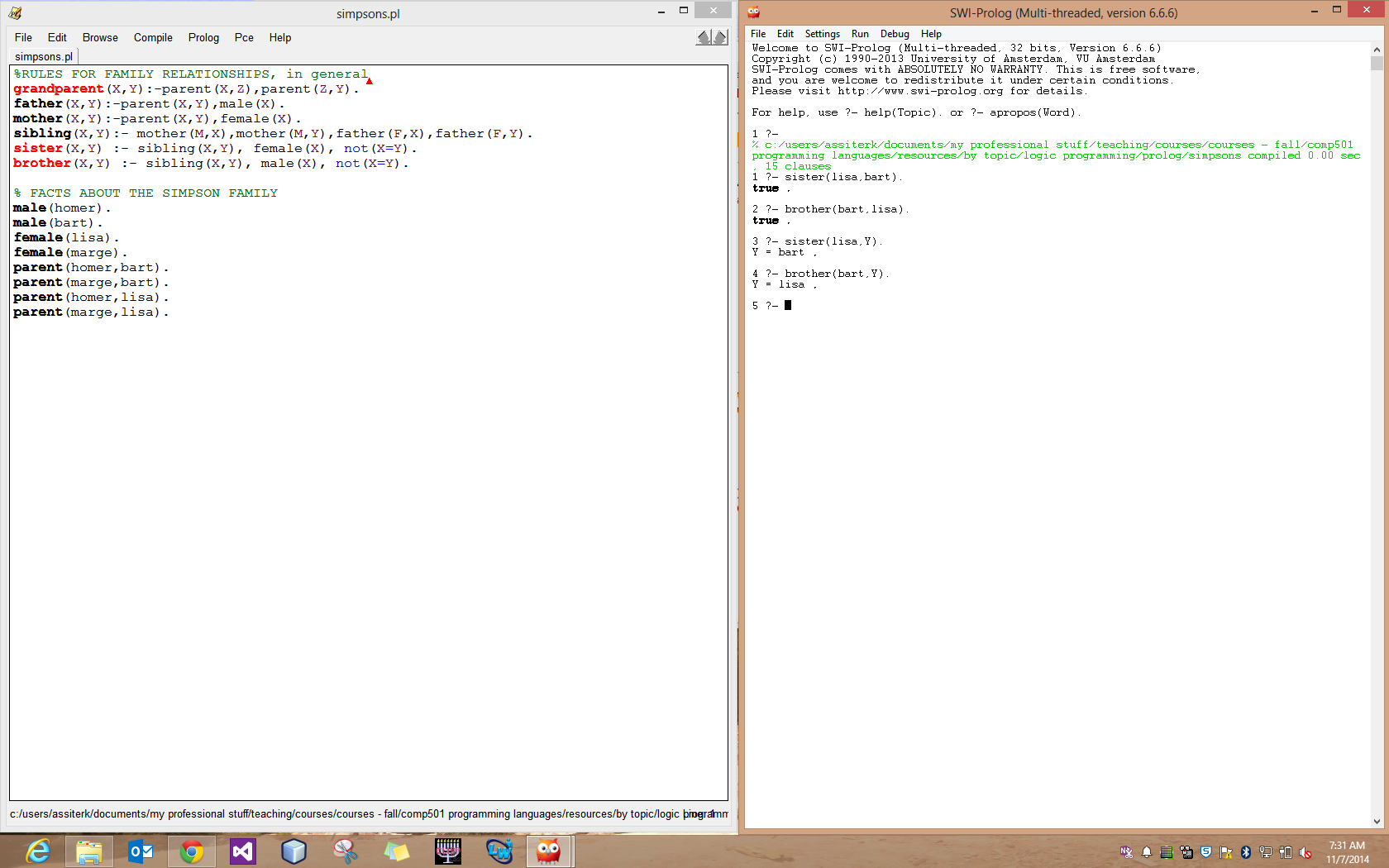
# PROLOG Example

This is **NOT** an example of an expert system, it is just meant to demonstrate the syntax and semantics of prolog. Included here is a screen shot of 2 windows with:

1. Source file containing rules and facts



1. SWI-Prolog command line (on the right), shown after the source file has been compiled and a few queries have been posed against the knowledge base.



# Evaluation

|  |  |  |
| --- | --- | --- |
| *Area* | *Requirement* | *%* |
| *Application* | *Reference list from idea-research* | *3* |
| *Originality* | *8* |
| *Usefulness* | *8* |
| *Complexity* | *10* |
| *Review idea with Instructor* | *2* |
| *Implementation* | *Facts* | *20* |
| *Rules* | *34* |
| *Presentation* | *Overview* | *5* |
| *Demo of Queries* | *10* |

# References

The following is meant to be a starter-list of references; I assume you will add to this list as you explore expert system ideas.

|  |  |
| --- | --- |
| [1] | Wikipedia, "Expert System," [Online]. Available: http://en.wikipedia.org/wiki/Expert\_system |
| [2] | Association for the Advancement of Artificial Intelligence, "AITopics," [Online]. Available: aitopics.org |
| [3] | V. Zwass, "Expert Systems," [Online]. Available: http://www.britannica.com/EBchecked/topic/198506/expert-system |