CSC 425/525 Programming Assignment #2
Due Date: Wednesday, March 1

Remember that undergraduates are permitted to work in groups of up to 3.

The goal behind this assignment is to learn what a rule-based (production) system is and how to use a rule-based language. There are three languages that you might try, all of which are freely available. Clips or Jess are recommended although you could also try Prolog. For Clips, go to http://clipsrules.sourceforge.net/. For Jess, go to http://www.jessrules.com/. You *may* be able to find a working Prolog compiler/environment on the Internet although that is not guaranteed. One location to check is http://www.thefreecountry.com/compilers/prolog.shtml. You can also consider Ops5 but it is not recommended.

The system you create will be a rule-based (production) system. Your system will consist of rules and starting pieces of knowledge (facts). You need a minimum of 75 rules/facts combined with at least 25 rules. Also, make sure that your rules do not simply map conditions to final conclusions. Build your system in such a way that it performs chaining to go from initial state to final conclusion. For an average case, your chain should be a minimum of 4 rules long. Some special cases can be shorter.

Select a domain that you are knowledgeable about. The knowledge does not have to be complete or 100% correct, but your system should draw useful conclusions. Example domains including reasoning about factual information such family relationships, animals, some form of diagnosis (medical, mechanical, program debugging), or goal-directed reasoning such as meal selection, daily planning or course scheduling. If you have questions about a possible domain, please ask the instructor.

Note that none of these languages have useful interfaces. In Clips, you can "turn dribble on" which sends all output both to the screen and a textfile. For Prolog output, you may have to obtain screen captures. Do not worry about creating an interface or fancy I/O.

Design the rules of your system. Then, implement, debug and run your system. If you find your system making a lot of logical mistakes, work on the logic. Run your system on a number of different inputs (initial facts) to obtain a variety of conclusions.

Hand in:

1) the source code of your program
2) several run traces (at least 5) each of which proves or performs a different problem within the domain
3) a short report describing your domain and knowledge, any difficulties you had in implementation, problems that your system has (for instance, specific problems that it could not solve correctly) and any enhancements that you feel should be made if you were going to spend more time on it.