**CS471 (Yoshii) Fall21: Final Project Descriptions**

**This is your final project in lieu of the final exam; therefore, it will be**

**due on Saturday of Week 16 to Cougar Courses.**

**No late work will be accepted. Do NOT email it to me.**

**Not following the instructions will result in points taken off so please read carefully.**

**Requirements:**

* **You must do this by yourself.**
* **You must write in your own words. No copying whatsoever is allowed.**
* **You must choose one of the following topics in which you are interested the most.**
  1. **Games/puzzles**
  2. **Knowledge-based systems**
* **You must follow the program specification exactly or you will not get any points.**
* **Total 30% of your grade.**

**Learning Objective:**

* **To apply what you have learned to a real problem.**
* **To implement the solution.**
* **To think about extensions to your system.**

**[Your job is to show me that you can do these on your own.]**

**Format of Your Report:**

* **Required on the Cover Page:**
  1. **CS471 and Title of the project**
  2. **Your name**
  3. **State of your system (Completed? If not, explain in detail.)**
  4. **The Check List**
* **Clearly number and title each Section of your report (Section Header) as seen in this file below. Try to write your answer after each prompt as given in this file.**
* **Submit a MS Word file (or .rtf) of the report and the source code files to Cougar Courses.**

**=============================================================**

**Check List (This must be checked off and included in your Cover Page):**

1. **\_\_\_ Did you follow all the requirements in implementing the system?**
2. **\_\_\_ Did you create one Word file of your report with the cover page and section headers as specified?**
3. **\_\_\_ Did you answer all questions per section?**

==============================================================

**The 2 choices are found below. Each is 30pts total.**

**Choice 1: Puzzle System [5pts per section]**

Your task is to implement a puzzle agent using **C++.** **(refer back to fwdc.cpp)**

This time, you have to solve the **3 pegs 3 disks** **Towers of Hanoi using A\***.

**Minimum System Requirements:**

* **DO NOT copy a program from some site! This is cheating.**
* It must be testable using g++ on empress.csusm.edu.
* It must have lots of useful comments.
* The output from the program should indicate the same amount of information as **fwdc.out** I had given you.

**Section 1: Analyzing the Problem Space**

1. How many unique states (include all possible whether reachable/legal or not)?
2. Show the equation on how you came up with this number.
3. List all possible disk move actions/operators (what goes where):
4. Therefore, the Branching factor: <= ??

**Section 2: Drawing the Problem Space**

1. Draw all states that are reachable/legal, and draw all possible arrows between them to create a graph. No node shall be duplicated (i.e. one node per state).
2. Label the arrows with moves (you may use a “legend” to make it easy to label links).
3. Mark the initial and goal states.

**Section 3: Designing the Evaluation Function**

**Assume that g = number of disk moves so far and**

**h = estimate of how many more moves a given state**

**f = g+h is the goodness**

1. Give the equation for the estimate function **h** (i.e. how do you guess the # of moves?)
2. Give an example **h** value and **g** value for **3** of the states in the above Problem Space.
3. Defend your decision for the **h** function:
4. Is this a good estimate of how many more moves are required?
5. Is this h admissible? Explain.

**Section 4: Implementation (Source Code is submitted separately)**

1. Must have functions dedicated to do the following. Write the name of the function next to each:
   1. Basic framework of puzzle Name: File:
   2. Generating the new states Name: File:
   3. Choosing the next state to expand Name: File:
2. What data structure (type and name) did you use for each node/state?
3. Give a picture of it with examples values.
4. What data structure (type and name) did you use to store all the Frontier nodes/states?

**Section 5: Testing and Results**

1. Test your program thoroughly and make sure the output matches your expectations.
2. Include here the test results (screen snapshots/recorded script compiled with g++).
3. Your analysis of the test results:
4. Did it work as expected? If not, explain.
5. How many disk moves did it take to reach the goal?
6. Is that optimal? Or a person can do with fewer moves?

**Section 6: Ideas for Adding Machine Learning**

1. **Give one way to incorporate machine learning into playing puzzles in general**

* What part of the program will the puzzle program update to improve itself?

1. **Then answer the following questions for your own project program:**

* How and when would this (from #1) learning happen?

(Being advised by a human as it plays?

Automatically by observing an outcome?) Be specific.

**Choice 2: Knowledge-Based System [6pts per section]**

Your task is to construct a **mini-rule-based expert system using Prolog for Diagnosis or Classification/Selection** in the domain of your choice. It must be useful. Check with me if you are in doubt. **(refer back to ESexample.pl)**

**Minimum System Requirements:**

* Must have at least **15 “main”** rules, and at least **7** **sub-goal** rules. (By “main” I mean rules such as “choose” in the ESexample).
* Must have **confidence/certainly factors** added as comments above each fact and rule.
* Must have lots of useful comments describing the rules and facts.
* Implement the system in Prolog (must be testable using gprolog on empress.csusm.edu).

**Section 1: Purpose of the Expert System**

1. Describe your useful domain and state wither it is for Diagnosis or for Classification/Selection.

e.g. Medical diagnosis for lung related diseases based on symptoms

e.g. Classification of birds based on physical characteristics

e.g. Selection of cars based on customer needs

1. State the goals/conclusions that the system reaches. State the user inputs in detail:
2. List of all possible conclusions:
3. List of all the information you need from the user with possible values:

**Section 2: Implementation (Source Code is submitted separately)**

1. State the **types** **of facts** (if any) available in the knowledge base.
2. Describe the **types of rules** the system uses. For example, for the **ESexample**, describe what choose rules are for, what reddish rules are for, etc.

**Section 3: Testing and Results**

1. Test your program **very** thoroughly and make sure it matches your expectations.
2. Include **3** example runs of the program (screen snapshots/recorded script), inserted into the document:

* Test 1: should show what happens when there is no answer.
* Test 2: should arrive at an answer using the first rule.
* Test 3: should arrive at an answer after some backtracking (goes down a path and backs up because a sub-goal fails).

Do not use ; to get multiple answers. Just one answer is enough.

**Section 4: Confidence Analysis**

1. What is the confidence you have in Test 3 answer from Section 3?

* Use the Mycin's method to calculate the confidence.
* Show all calculation steps.

**Section 5: Ideas for Adding Machine Learning**

1. **Give one way to incorporate machine learning into rule-based systems in general**

* What part of the program will the program update to improve itself?

1. **Then answer the following questions for your own project program:**

* How and when would this (from #1) learning happen?

(Being criticized by a human user?

Through Big Data analysis?

Automatically by observing an outcome?) Be specific.