**CS471 (Yoshii) FALL22: Final Project**

**And ABET Assessment**

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

**due on Saturday of Week 16 to Canvas!**

**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 follow the program specification exactly.**
* **Total 30% of your grade.**

**Learning Objective:**

* **To apply what you have learned to another 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: [Correct formatting and submission is worth 5pts]**

* **Required on the Cover Page:**
  1. **CS471 and Title of the project**
  2. **Your name**
  3. **Your major (CS?CIS?SE?)**
  4. **The Check List (see below)**
* **Clearly number and title each Section of your report (Section Header) as seen in this file below. Write your answer after each prompt as given in this file.**
* **Submit a MS Word file (or .rtf) of the report and the output file(s) and the source code files to Canvas.**

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

**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?**

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

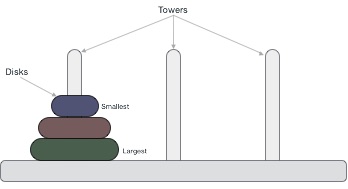
**Requirements are found below. 30pts total.**

**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\***.

**The initial state is having all 3 disks on Peg 1.**

**The goal state is having all 3 disks on Peg 3.** 

**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. (Show the frontier, the expanded node, with g,h,fs each time)**

**Section 1: Analysis of the Problem Space (6.1)**

1. List all possible disk move actions/operators (what goes where) – note that only the disks at the top can move to another peg:
2. Therefore, the Branching factor: <= ??
3. Draw all states that are reachable/legal, and draw all possible arrows between them to create a graph. No node shall be shown twice (i.e. one node per state).
4. Label the arrows with moves (you may use a “legend” to make it easy to label links).
5. Mark the initial and goal states. Mark the shortest path from Initial to Goal.

**Section 2: Designing the Evaluation Function for A\* (6.2)**

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

**h = estimate of how many more moves from 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 why.

**Section 3: Implementation (Source Code is submitted separately) (6.3)**

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 all 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 an abstract picture of it with examples values.
4. What data structure (type and name) did you use to store all the Frontier nodes/states?

**Section 4: Testing and Results (6.4)**

1. Test your program **very** **thoroughly** and make sure the output matches your expectations.

Produce **a .txt file of test results** (recorded script compiled with g++).

1. Your analysis of the results:
2. Did it work as expected? If not, explain exactly where it failed. (This includes correctness of all the intermediate outputs)
3. How many disk moves did it take to reach the goal?
4. Is that optimal? Or a person can do with fewer moves? Check against your graph.

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

1. **Give one way to incorporate machine learning into your program to make it better?**

* What part of the program will the puzzle program update/adjust to improve itself?
* How and when would this learning happen in your program?

(Being advised by a human as it plays? Exactly at what point in the program?

Automatically by observing an outcome? Exactly at what point in the program?)

Be very specific **using an example.**