## Assignment - In progress

| | |
|---|---|
| **Title** | Project 1 |
| **Due** | Sep 14, 2012 11:55 pm |
| **Status** | Not Started |
| **Grade Scale** | Points (max 100.0) |
| **Modified by instructor** | Aug 27, 2012 4:59 pm |

**Instructions**

**Goal:** The goal of the project is to learn about the close relationship between learning and problem solving. On one hand, results of learning often help us solve problems. On the other, we often learn what we need to address some problem. The task here is to address A:B::C:x? analogy problems from Miller's test of intelligence.

**Deliverables:** You will be given three visual reasoning problems. Your goal is to write a propositional representation for each of these problems and a program that can solve these problems based on that propositional representation. As such, you will deliver three items:

1. Propositional representations (a text file) for all three problems.
2. A computer program that loads those propositional representations and solves the problems. This program must be written in either Java, Python, or C#.
3. A report that explains and justifies the design of the architecture and algorithms of your program and the experiments you conducted with it.

We will supply you with the visual representations of the three problems (see the attachments to this project), but these are only for your benefit - yuor program will not read them. Instead, you should use these visual representations to construct your own propositional representations which will act as the input. The output, in turn, is your program's answers to each of the three problems.

All these deliverables must be turned by the due date.

**Fourth Problem:** After the due date, a fourth problem will be supplied. You will have one week to return a propositional representation of this fourth problem. The goal here is for your original project to be sufficiently general that it can approach new problems that are structured according to your propositional representation schema. You will not be able to revise your source code after receiving this fourth problem. Your propositional representation for the fourth problem must be turned in by one week after the project's due date.

**Structure:** Your deliverable should be a .zip file with the name (yourfirstname)(yourlastname).zip. The contents of the .zip file should follow the following structure:

- A report named (yourfirstname)(yourlastname)_Project_1 describing the rationale, architecture, and algorithms of your program in accordance #3 above. This document can be a .pdf, .doc, or .docx file.
- A folder named Source Code that contains the entire source code of your project. Make sure to include any libraries or files that would be necessary to recompile the project.
- A folder named Representations that has the Propositional Representations of your problems in .txt format. Each representation should be the project number followed by the problem

number; for example, Problem 2 of Project 1 should be named 1-2.txt.

- An executable file named (yourfirstname)(yourlastname)_Project_1 in the root folder of the .zip file that, when run, will solve the problems in the Representations folder.
    - If your project is written in Java, this should be (yourfirstname) (yourlastname)_Project_1.jar.
    - If your project is written in C#, this should be (yourfirstname) (yourlastname)_Project_1.exe.
    - If your project is written in Python, this should be (yourfirstname) (yourlastname)_Project_1.py.
- It is fine to include other folders in the .zip file if necessary (for example, a lib folder that the .jar file accesses) so long as the above two folders and two files are present.

**Running the Program:** When executed, your program should follow the following guidelines:

- The program must automatically begin to solve the problems when run, without requiring input parameters. You should hardcode the filenames for your representations into the program.
- Your program should be prepared to answer all four problems -- your fourth problem representation will be placed in your Representations folder as 1-4.txt. Ideally, your program would detect whether 1-4.txt is present and only solve it if so, but it is acceptable to include a placeholder 1-4.txt file that will be overwritten once your fourth representation is supplied.
- Your program can run with either a GUI or text interface.
- Your program must display some visual sign of ongoing progress. These algorithms can take quite a while to run, so we need some way of knowing whether your program is still working or if it has locked up.
- Your program must display the answers to all solved problems in clearly-readable format at the conclusion of the program's run. For example "Problem 1: A". Make sure to use the same letters used in the problem to give your responses.

**Report:** The report should contain a detailed description of how the algorithm operates, including your design rationale for designing it the way you did. Make sure to comment on the algorithm's efficiency: how much extra time will be involved if the problem complexity expands?

**Grading:**

- 20% for correctly solving the first problem.
- 20% for correctly solving the second problem.
- 20% for correctly solving the third problem.
- 20% for correctly solving the fourth problem.
- 20% for the written report, including an evaluation of the efficiency and generalizability of the algorithm.

A sample propositional representation is included. Note that it is not necessary for you to use this representation, or even anything resembling this representation -- you are free to establish semantics and structures for your representation as you wish, but remember you will be required to create a representation of a problem you have not yet seen after delivering your source code.

If you are unsure about the structure of your deliverable, feel free to visit David Joyner's office hours to get confirmation that your deliverable is properly structured.