**Comp 445 Project 1 Fall 2015**

**Artificial Intelligence 100 Points Dellinger**

Due: 11:55 PM, October 12, 2015

**Programming Requirements**

For this project you may work in teams of two or three people. Some variation from the steps below may be possible; if you have an outside-the-box idea, however, you must clear it with me in advance.

Apply one or more of the algorithms discussed in class to the solution of a deterministic one-player puzzle. Some examples are Sudoku, generalizations of the eight puzzle and the fifteen puzzle, the Cracker Barrel puzzle (and/or puzzles like it), the *n*-queens problem, and the word transformation puzzle (minimum number of letter swaps, letter replacements, letter insertions, letter deletions, etc., to transform one word into another, using words from a specific set as intermediate states).

As a general rule, your chosen project should allow multiple “cases.” For instance, the *n*-queens problem can be attempted many different ways (with different choices of *n*), and so you could choose a single algorithm to apply to several different values of *n*. On the other hand, if you chose to work with the 8-queen problem specifically (which has only one “case”), you should choose several different algorithmic approaches to apply to it.

You will need to submit your basic idea, including the group you are working with (if any), for approval by one week after this assignment is given. Puzzles that are overly trivial or algorithms that are trivial or intractable may not be accepted. (For instance, it’s fine to write a Sudoku solver; it is *not* fine to try to solve Sudoku through pure unguided brute-force search.)

Having chosen a problem, develop a class (or collection of classes) in C++ that implements your chosen problem. You should then write a “solver” class (or collection of related classes) that, on being given an instance of the problem, will attempt to solve it using your chosen algorithm. If successful, it should present the solution to the problem in a human-readable fashion. (You may find it easier to provide a GUI for your application than to do this in the console, but you may choose any approach as long as it’s possible for me to clearly understand what I’m seeing.)

Your final submission should demonstrate a range of different cases (either different starting problems, applications of different algorithms to a single problem, or both). The project must be thoroughly tested; I should not be able to break it if I assign my own problem cases. Include a brief README file explaining how to use and interpret your program.

You may discover that your chosen algorithm(s) cannot solve certain classes of problem in a reasonable amount of time. This is not a problem; see below.

**Writing Requirement**

Investigate the technical artificial intelligence literature to find articles regarding approaches to solving problems like the one you are attempting. (You may not be able to find articles that apply your specific algorithm to your specific puzzle – if not, great! Find similar attacks on your puzzle or uses of your algorithm on similar puzzles.) You should find at least two closely related sources; you may find resources such as AAAI, JAIR, or Google Scholar helpful in this task.

Write up a paper summarizing your chosen problem, the results you found in your research and how those informed your work, and your own experimental results. If you found that your approach could not solve certain classes of problems, include that in information in the results. Cite any resources used, and include a bibliography at the end. (Your references should follow IEEE style, as [here](http://www.ieee.org/documents/style_manual.pdf).) The paper will be evaluated on both technical content and grammatical accuracy. The paper should be at least two pages long (single spaced, not including bibliography) and should not exceed four pages (again, not including bibliography).

**Presentation Requirements**

Prepare a brief (5-8 minute) presentation describing your problem, your algorithm, your research, and your results. (Given the allotted time, this should be a fairly high-level discussion.) You may use visual aids such as Power Point, and you must demonstrate your program.

**Submission**

Zip the document and your source code, along with any data used or produced by your source code, into one file and upload it to MyGCC before 11:55 on Monday, October 12. Be prepared to present your work as a presentation in class on the following day.

**Grading Rubric**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Details** | **Comments** | **Points Earned** | **Points Possible** |
| Programming Requirement (60%)  *(Outcomes 1, 3)* | Project compiles without errors. |  |  | 5 |
| Project displays good programming style: appropriate comments, code divided into classes in logical ways, problems and solver appropriately separated, included README file, etc. |  |  | 10 |
| Program correctly implements the underlying problem to be solved. |  |  | 15 |
| Program correctly implements the algorithm(s) to use on the problem. |  |  | 15 |
| Program demonstrates a range of test cases for different problems/algorithms. |  |  | 15 |
| Written Requirement (30%)  *(Outcomes 2, 4)* | Paper is well-structured and grammatically sound. |  |  | 5 |
| Paper includes an IEEE-formatted bibliography with at least two sources. |  |  | 5 |
| Paper summarizes existing research clearly. |  |  | 10 |
| Paper describes the problem to be solved, the algorithm used, and the results. |  |  | 10 |
| Paper is of an appropriate length. |  |  | 5 |
| Presentation Requirement (10%)  *(Outcome 4)* | Presentation is smooth, audible, and of appropriate length. |  |  | 5 |
| Presentation summarizes the problem, the approach, and its results clearly. |  |  | 5 |

Each category will be graded as follows:

* 0%: Missing, not done, drastically wrong.
* 20%: Largely incorrect or missing.
* 50%: Some correct work, but heavily flawed.
* 80%: Largely correct work, with some minor issues.
* 100%: Free of all but the most trivial errors.