CSCI 111 Introduction to Artificial Intelligence
Group Project

You may work in groups of at most 3 students. Choose from among the following items or propose a topic of similar complexity. You are expected to present your project in class at the end of the semester.

1. Building scenarios for autonomous vehicles

   Autonomous vehicles undergo testing through platforms that simulate task environments. Simulation platforms such as esmini (https://github.com/esmini/esmini) and CARLA (http://carla.org/) use the OpenDRIVE (https://www.asam.net/standards/detail/opendrive/) and OpenSCENARIO (https://www.asam.net/standards/detail/openscenario/) standards. You are tasked to set up a task environment for a self-driving vehicle using any of these simulation software platforms (esmini or CARLA). Much of the work involved for this project is setting up and understanding the software and then specifying a nontrivial scenario to be demonstrated in class.

   Deliverables: Slide presentation explaining how scenarios are built in the platform chosen and instructions on how to setup scenarios and run simulations.

2. Search trees and algorithms for the 8-puzzle

   Consider the 8-puzzle problem described in class. Demonstrate states, search trees, and search algorithms (at least 2 algorithms; e.g., BFS and A*-Search). Solve the problem for two nontrivial initial configurations. You may write a program to carry out the search.

   Deliverables: Slide presentation explaining the 8-puzzle problem and solutions to the two problems/initial configurations and programs if applicable.

3. Evaluating Machine Learning models

   Select a dataset from UCI (https://archive.ics.uci.edu/ml/datasets.php) or Google (https://datasetsearch.research.google.com/), formulate a machine learning problem (supervised or unsupervised), and build and evaluate two models (different methods) that solve the problem. Clear the dataset with your instructor before you proceed.

4. Converting propositions to CNF (conjunctive normal form)

(You may choose this option ONLY if you are working alone)

Write a a LISP function called CNF that takes a propositional logic expression and returns that expression in Conjunctive Normal Form. Conjunctive Normal Form means a conjunction (**and**-ed) of disjunctions (**or**-ed) of literals (a symbol or the **not** of a symbol). Assume that the input expression is fully parenthesized and written in infix form. The connectives are specified by the following keywords: **and**, **or**, **not**, **implies**, and **equivalent**.

The connectives are specified by the following keywords: **and**, **or**, **not**, **implies**, and **equivalent**. You may assume that the propositional symbols do not clash with these words. The following are examples of calls to the CNF function, illustrating some possible expressions:

```
(CNF ‘(simple))
(CNF ‘(not hard))
(CNF ‘(a implies b))
(CNF ‘((a or b) or c))
(CNF ‘((not (x or b)) equivalent ((not y) and (not (z implies a)))))
```

The CNF expression returned will be in a different form, for simplicity. The function will return a list of lists of literals. The atoms in the returned expression are just symbols and the keyword not. For example, the list (which is the output this assignment requires)

```
( (a b (not b)) (x) (r (not s) y) ((not z)))
```

means:

```
( (a or b or (not b)) and (x)  and (r or (not s) or y)  and ((not z)) ).
```

The following stages are suggested in the conversion:

1. Eliminate all equivalences and implications, so that only **and**, **or**, and **not** operations remain.
2. Apply De Morgan’s law so that all not operations are applied to literals only.
3. Apply grouping to achieve conjunctive normal form.

Deliverables: Programs written in LISP and test code (your instructor will provide test files)