BLG435E, Artificial Intelligence, Fall 2018-2019
Assignment #1

Submission Type: An archive file including a softcopy report and the source codes for Q3 to be submitted using Ninova. Solutions for Q1 and Q2 must be **handwritten and scanned/photographed** in your report. Note that each student must work individually for this assignment. Team work is not accepted!

Q1 (15 pts) Select one of the following agents (show which one you have selected) and develop a PEAS description of the task environment:

(a) Domestic service robot
(b) E-mail sorting application based on preferences
(c) Public transportation recommendation system
(d) Activity recognition and anomaly detection software agent in an airport

For the agent you have selected, characterize the environment according to the properties of the environment (observability, dynamism, etc.) and determine the appropriate type of the agent architecture with reasonable arguments.

Q2 (15 pts) Prove that if a heuristic is consistent, it must be admissible. Construct an admissible heuristic that is not consistent.

Q3 (70 pts) In this section, you are asked to solve the simplified peg solitaire game with the standard (English) board. In this problem, the board is filled with the pegs except for the central hole. Each peg can jump orthogonally over an adjacent peg into an empty hole and then the jumped peg is removed. Different than the original game the objective in this

assignment is making valid moves until no valid move exists. Number of the pegs in the final state is irrelevant in our objective definition. To clarify the problem, an example illustration scenario of the game is given in the following figure.

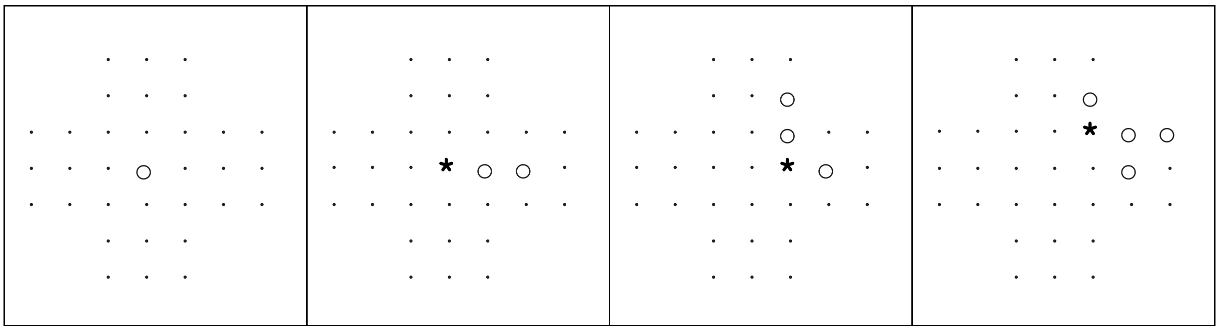

Figure 1: Dots and empty holes are represented by dots and circles, respectively. Stars represent the peg that is moved in the last turn.

Objective: Making valid moves until no valid move exists.
The number of remaining pegs will **not** be considered as an objective in our problem.
Valid move: Each peg can jump orthogonally (left, right, up or down) over an adjacent peg into an empty hole and then the jumped peg is removed.

(a) Formulize this problem in a well-defined form and present your state and action representations in detail.

(b) Run tree search versions of both Breadth-first search (BFS) and Depth-first search (DFS) and analyze the results in terms of:

- the number of nodes generated
- the number of nodes expanded
- the maximum number of nodes kept in the memory
- the running time.
- the number of pegs in the found final state

If any of the algorithms does not last, please specify the reason.

(c) Run A* algorithm to find the **minimum** number of moves to reach a final state (where no valid move exists). Implement your algorithm with two different admissible heuristic functions and give a detailed analysis of the results in your report in terms of:

- the number of nodes generated
- the number of nodes expanded
- the maximum number of nodes kept in the memory
- the running time.
- the number of pegs in the found final state

(d) If the objective is changed to finding a state where only one peg remains, how will it affect the running time of BFS, DFS and A* algorithms? Explain it briefly, you do **not** need an implementation for this variation.

Important: In Q3, your solution can rely on existing BFS, DFS or A* algorithm implementations such as the ones provided for “Artificial Intelligence: A Modern Approach” book. However, you need to explain how the algorithms work in this problem and perform the requested analyses above with sufficient explanations in your report. **Code usage without relevant references will be considered as Plagiarism.**