**CSINTSY** Release Date: February 1, 2023
MCO1 Specifications: MazeBot Submission Date: March 4, 2023

## Learning Outcomes

This project is a venue for the students to achieve the learning outcomes below:

**LO1.** Design and evaluate informed search algorithms and knowledge representations for problem solving
**LO2.** Collaboratively build systems that consider a number of paths or strategies in order to improve its performance in achieving its goal in less amount of computing time, or by some other metric of performance
**LO4.** Perform comparative analyses of algorithms for problem space search and machine learning using real-world datasets
**LO5.** Articulate ideas and present results in correct technical written and oral English

Students will demonstrate their skills in developing and evaluating the artificial intelligence of a bot. They are going to apply a tree search algorithm to make their bot smart. Then, the behavior of their bot must be examined carefully to decide whether to consider the bot intelligent or not. They will also write a report explaining very clearly and convincingly why they think the bot is intelligent or unintelligent.

## Maze

The setup is an $n \times n$ maze, where $n$ is any number from 3 up to a maximum of 64. A maze configuration can be defined as a text file called **`maze.txt`** in the following format:

```
n
XXX
XXX
XXX
```

The first line of the text file contains a single integer $n$, the size of the maze. This is a followed by $n$ lines containing $n$ characters each (represented as Xs in the definition above). Each X may be one of the following:

| **Symbol** | **Description** |
|---|---|
| # | wall |
| . (period) | an empty space |
| S | starting location of the bot |
| G | target location of the bot |

You may assume that the maze **always** contains exactly one S and exactly one G, otherwise it is an invalid maze.

Here is an example of an actual maze definition:

```
5
....G
.####
...#S
.#.#.
.#...
```

**The Bot**

Given a maze configuration, you are to implement a bot that must find its way to the goal. The bot can only step on a space, and its movements are restricted to go one step up, down, left, or right.

The simulation of the search must be illustrated to show which paths were explored before getting to the goal. The chosen path must be highlighted differently.

**Expected Program Behavior**

You are to implement a program with the following behavior:

- Once executed, the program loads and displays the maze defined in **`maze.txt`**. You're free to choose the way the look of the maze as long as it is easily interpretable. However, don't spend too much time and effort making the maze look good at the expense of the more important parts of the project.
- The user can initiate the search. After which, the bot will find a path from the starting point (S) to the target location (G). The program should display (a) the final path taken by the bot, , (b) all the locations that were explored by the implemented algorithm before arriving at the solution, and (c) the total number of states explored before the solution is found (or not found).
- The program should also display the **order** in which the locations were explored by the algorithm. It is up to you how to want to display this. Some options are:
    - Put a number to show the ordering
    - Use an animation showing the step by step exploration of locations by the algorithm (either timed or through a button to advance to the next step).
- If there is no path to the target location, the program should state that a solution cannot be found, and must **not** crash.

Note the following things:

- You can use any programming language.
- Resulting program can be text-based, web-based or could be a desktop application. But it ***must*** run on a Windows-based platform. Regardless of what you choose, make sure you comply with **all** features listed above.
- You may assume that **`maze.txt`** will always be valid according to the definition above. No need to write checks for this.
- To get full points for this project, your bot should ***at least*** be complete (i.e., finds a solution if it exists).
- No need to create interfaces for:
    - Creating / editing a maze (maze configurations can just be specified through the text file, no need to have an editor)
    - Restarting the search after it is completed (we can just restart the program if we want to restart the search).

**Report**

In addition to the above program, you are to write a report containing the following sections:

**I. Introduction**

- In paragraph form, describe the task and give a brief overview of the algorithm used by your bot.

**II. Program**

- Write the step by step instructions in detail on how to run the program and use its features.

**III. Algorithm**

- Present the **pseudocode or flowchart** of how the bot works, and describe in detail the algorithms and rules used by the bot. Make sure that these are **well-defined** and **easy to understand**.

### IV. Results and Analysis

- Discuss what situations the bot can handle. Explain why the algorithm is able to handle these cases.
- Discuss what situations the bot cannot handle, or what situation the bot performs poorly. Explain why these situations cannot be handled by the bot. Point out which part of the AI made it fail.
- You may include screenshots if need be.

### V. Recommendations

Based on the analysis of the performance of the bot, point out the weaknesses of the bot. Identify and explain possible ways to address these weaknesses.

### VI. References (if any)

- Use APA citation format

### VII. Contributions of Each Group Member

- Explain in detail the contribution of each group member in the completion of the project.

The report should also include a title page containing the members of the group.

There is no minimum number of pages for the report, but keep in mind that you will graded based on its **substance**, **readability** and **completeness**.

## Deliverables

You are to submit the following through AnimoSpace on or before the deadline set by the instructor:

a. The executable program (ready to run). This should be in a folder named **`app`**. If there are additional files needed for the program to run, make sure to include them.
b. The complete source codes used for the program. Put these in a folder named **`source`**.
c. A .pdf document containing the report. Name this file **`report.pdf`**.

These are to submitted as a single zip file with a filename of **`MCO1_<surname 1>_<surname 2>_<surname 3>_<surname 4>.zip`**. Please make sure that you comply with the naming conventions and format of the submission.