# ECM2423 - Coursework exercise

**Deadline: Thursday 16th March (midday, Exeter time)**

Lecturer: Dr. Ayah Helal (a.helal@exeter.ac.uk)

This assessment is worth 40% of the overall module assessment. This is an individual exercise and your attention is drawn to the College and University guidelines on collaboration and plagiarism, which are available from the College website.

**Referencing, and academic conduct** You are required to cite the work of others used in your solution and include a list of references, and must avoid plagiarism, collusion and any academic misconduct behaviours.
`https://vle.exeter.ac.uk/course/view.php?id=1957`

**Coursework submission** The submission of this coursework is via **eBART**. The code needed to answer the questions must run without errors. Make sure that your code is clear, commented, and that your answers are clearly labelled.

In **eBART**, you should submit a **single compressed (zipped) folder** which contains all the relevant files for this coursework exercise. The compressed folder can contain **either** a PDF file containing your answers and files containing your source code (Python scripts, Java files, C files, etc) **or** Jupyter notebook containing both code and answers if that is what you prefer using.

**Marking criteria** The question you will find below account for 75 marks. The question is further divided in to sub-questions to guide your answers, and specific marks are provided for each of the sub-questions. Additionally, for each sub-question I have clearly defined what the deliverable should be, so that it is clear what you should produce for your answer. A further 25 marks will be awarded according to the following three criteria:

1. code documentation and comments, with all the relevant information needed on how to run the code (including the names of scripts so that they can be clearly linked to a specific question): **5 marks**
2. clear presentation of your answers. **5 marks**
3. further experimentation, depth of analysis or discussion. **15 marks**.

The sum of the marks that you can obtain in this coursework exercise cannot exceed 100. In general, all questions will be marked according to the following criteria:

1. For questions which require code (question 1.2 part 2, and 3; question 1.3 part 2): have you correctly implemented the algorithm? Does your implementation solve the problem as required in the question? Does your code run without errors? Is the code clearly structured?
2. For questions requiring a written answer (question 1.1; question 1.2 part 1, and 4; question 1.3 part 1, 3, and 4): is your answer clear, coherent and well-structured? Does it give the best answer to the question? If applicable, have you made the most of possible figures and plots to support your answers, and are those figures and plots appropriately designed and clear? Is your answer succinct and does it only contain information relevant to the question? Is your answer concise and to the point ?

## Question 1 | Maze Solver.

Lets consider a maze which has height $h$ and width $w$. The maze is presented as '#' '-', where '#' is a wall, and '-' is a path. The goal is to find the path from the start node, which will is the only path on the top line of the maze, to the goal which is the only path on bottom line in the maze.

```
# - # # # # # # # # # # # # # # # # # #
# - - - - - - - - - - - - - - - - - # #
# # # # # - # # - # # - # # # - # - - #
# - # # # - # - - # # - # # - - # - # #
# - # - - - # # # # # # # # # - # - - #
# - - - # - - - - # - # # - # - # # # #
# - # # # # - # - - - - - - - - - - - #
# - # # - - - # # # # # # # # # # - # #
# - - - - # - - # # - - - - - - - - - #
# # # # # # # # # # # # # # # # # # - #
```

Figure 1: Example of the easy maze.

### Question 1.1: Describe how you would frame the maze solver as a search problem.

In your own words, write a short description of how the maze solver can be seen as a search problem.

**(5 marks)**
**Deliverable:** written answer.

### Question 1.2: Solve the maze using depth-first search.

This question is further divided into parts. This is a guide to help your in the coding process. You do not need to submitted different versions of the code, but just the last updated version.

1. Briefly outline the depth-first algorithm.

   **(5 marks)**
   **Deliverable:** written answer.

2. Implement depth-first to solve the maze. The solution of the maze should print the locations in the map that the algorithm will take to reach from the start to the goal. This question can be solved using the file 'maze-Easy.txt".

   **(10 marks)**
   **Deliverable:** code and the path found to solve the maze.

3. Update your algorithm to calculate some statistics about its performance. The number of nodes explored to find the path, the time of the execution, and the number of steps in the resulting path.

   **(5 marks)**
   **Deliverable:** code and written answer analysis of the maze.

4. Update your algorithm to be generalised to read any maze in the same format. Then calculate the paths and statistics for all 'maze-Medium.txt"/ 'maze-Large.txt" / 'maze-VLarge.txt".

   **(10 marks)**
   **Deliverable:** code and written answer analysis of the different maze.

## Question 1.3: Suggest an improved algorithm for this problem.

This question is further divided into parts. This is a guide to help your in the coding process. You do not need to submitted different versions of the code, but just the last updated version.

1. Briefly outline another algorithm to solve the maze. That decision should be justified by what you expect to improve in the performance over depth-first search.

   **(10 marks)**
   **Deliverable:** written answer.

2. Implement the suggested algorithm to solve the maze. It should also calculate the same statistics about its performance as the previous depth-first algorithm. The solution of the maze should print the locations in the map that the algorithm will take to reach from the start to the goal.

   **(15 marks)**
   **Deliverable:** code and written answer analysis of the different maze and the path found to solve each of the mazes given.

3. Run your algorithm to calculate the paths and statistics for all given mazes which are ‘maze-Easy.txt”/ ‘maze-Medium.txt”/ ‘maze-Large.txt” / ‘maze-VLarge.txt” .

   **(5 marks)**
   **Deliverable:** written answer analysis of the different maze.

4. Discuss based on your results, analysis how the suggested algorithm performed better than the depth-first search algorithm.

   **(10 marks)**
   **Deliverable:** written answer to analysis your result.