Project Brief

Study Unit	ARI2101 – Fundamentals of Automated Planning
Marks	30% of study unit assessment
Deadline	Monday 12th January 2026 – 15:00hrs CET

Read the instructions carefully, thoroughly and completely before attempting this assignment. Failure to comply with the requirements can result in unnecessary loss of marks or potentially disqualification.

This project needs to be completed as a **group**. Make sure you declare the members of your group beforehand. Each group should be made up of at least **2 students** and **not more than 3 students**.

Problem Description

Soko the robot works in a maze-like warehouse where boxes labelled A, B, C etc. need to be placed in their designated drop zone A, B, and C. Some of the boxes are placed in rooms behind locked doors, numbered 1, 2, 3 etc. where to open the door 1 the robot needs to have key 1, which would be located somewhere in the warehouse.

The objective of this project is to:

1. Analyse the maze to find the location of the robot, boxes, drop zones, doors and keys.
2. Find a plan that takes all the boxes to their designated drop zones.
3. Verify the plan by executing it and checking that it achieves the goal.

As part of this project you have been provided with a set of mazes to solve. Each maze is encoded as a CSV file where each line corresponds to the row in the grid, while the value between comas corresponds to the object in the cell, as follows:

- S – The location of Soko the Robot.
- W – A wall. The robot cannot traverse walls.
- B-<letter> - A box with the designated dropbox. B-A means the box that needs to be placed at the same location of drop zone A.
- Z-<letter> - A drop zone. Z-A means drop zone A.
- K-<number> - A numbered key. K-1 means Key which opens Door number 1.
- D-<number> - A numbered door. D-1 means Door number 1.
- Empty space – An empty cell where the robot can move that has nothing.

For example, the row:

```
W, , , ,S, , , ,K-1, , ,D-1, , ,B-A, , , ,Z-A, , ,W
```

Means a row with a Wall, 3 empty cells, Soko the robot, another 3 empty cells, key 1, 2 empty cells, door 1, 2 empty cells, Box A, 3 empty cells, Drop Zone A, 2 empty cells and a wall. Each file will have multiple rows, which together correspond to a 2D maze.

Actions

Soko the robot can perform the following actions. Each action costs 1 unit.

Action	Precondition	Effect
Left	There is no wall or door on the left.	The robot moves to the cell on the left.
Right	There is no wall or door on the right.	The robot moves to the cell on the right.
Up	There is no wall or door above the robot.	The robot moves to the cell above.
Down	There is no wall or door below the robot.	The robot moves to the cell below.
Take Key	The robot is in the same cell of a key.	The robot owns the key. The key is no longer in the cell on the grid.
Walk Left Through Door	There is a doorway on the left and the robot has the matching key.	The robot moves to the cell (doorway) on the left.
Walk Right Through Door	There is a doorway on the right and the robot has the matching key.	The robot moves to the cell (doorway) on the right.
Walk Up Through Door.	There is a doorway above and the robot has the matching key.	The robot moves to the cell (doorway) above.
Walk Down Through door.	There is a doorway below and the robot has the matching key.	The robot moves to the cell (doorway) below.
Lift Box	The robot is in the same location of a box and is not carrying any other box.	The robot is carrying the box.
Drop Box	The robot is carrying a box and it is at the drop zone location corresponding to the box.	The box is placed in the same location as the robot. The robot is no longer carrying the box.

Table 1: Possible Actions

Goal

The goal is that each box is in the designated drop zone identified by the letter of the box. So, Box A needs to be in drop zone A, Box B in drop zone B etc.

Implementation

Part 1 – Problem-Specific Solver

Using any programming language of your choice, develop a solver that reads the maze from a file, performs the search and finds the plan to the goal.

Input and Output

Your program must take the maze as input, search for a plan, and if successful, output the plan: a sequence of actions from Table 1. It must offer the user, through a menu-based system or the command line, the option to select different heuristics and search strategies.

Apart from the plan, as part of its output, it must also include the following information:

1. The search strategy and heuristic used (if any).
2. Time it took to find the plan.
3. Number of generated states.
4. Number of expanded states.
5. Length of the plan (number of steps in the plan).
6. Whether the plan is valid (see validation part below).

States and Actions

Choose an appropriate data structure to represent the state of the maze, the robot (and what it is carrying) together with the locations of the respective objects.

The successor states of a state need to be generated from the **applicable actions** according to preconditions and effects defined in Table 1.

Search Strategies

You need to implement all the following search strategies, from which the user can choose:

1. Breadth First Search
2. Greedy Best First Search
3. A* Search
4. Enforced Hill Climbing

Make sure your search algorithms have an efficient way of keeping track of visited states, so that the search does not loop repeatedly through states that there were already visited.

You should come up with **2 heuristics** that consider the location of the robot, location of the boxes and their drop zones, and locations of doors that might be in the path, and their respective keys if the robot doesn't have the key in the current state.

Hint: You can consider a heuristic based on either Manhattan distance or Euclidean distance but takes into consideration prerequisites of the actions, like having keys and carrying boxes.

Analyse the properties of each of your heuristics:

1. Why do you think it is adequately informative for the problem?
2. Is it admissible?

Plan Validation

After your search algorithm returns a plan, include a validation step which starts from the initial state and for each action in the plan, checks that it is applicable, and applies it to obtain the successor state. After the last action of the plan is applied, the last state should be the goal state (all boxes are at their respective drop zones).

Visualise the plan execution by displaying the maze in 2D, and step through the actions of the plan.

Part 2 – Generic Solver (PDDL)

PDDL Domain

The **PDDL domain** file should define object types for the respective objects that can be present, such as the robot, the cells through which it can navigate, doors and corresponding keys, boxes and corresponding drop zones.

It should also define the predicates that define the current state, such as the location of the robot and the rest of the objects, whether the Soko the robot is in a state that allows picking up a box, which box the robot is carrying (if any), and whether the robot has any keys. The relationship between doors and their corresponding keys, and boxes and their corresponding drop zones can also be expressed using propositional predicates.

Finally, it should define the actions (one for each action defined in Table 1), with their respective preconditions and effects.

PDDL Problem

Using the same maze file loader developed for Part 1, generate a PDDL problem file for each maze.

Each problem file should have:

- The objects and their respective types.
- The initial state, consisting of all adjacent cells that are navigable to each other, the location of the robot and the rest of the objects, and any predicates defining the relationship between keys and doors, and boxes and their respective drop zones.
- The goal condition, that each box is in the same location of its respective drop zone.

Test the domain and problem files using **at least one** of the online solvers.

For each problem, take note of the plan length, number of states generated, number of states expanded, and the time taken for the online solver to find the solution.

Evaluation

Plot the results on graphs or any other visual tools, to clearly compare the performance of the different search algorithms, heuristics and the online solver.

Analyse how the **plan length** and **number of states generated** differ between your domain-specific solver (and its different search strategies) and the domain-independent planner.

Deliverables

You should submit all deliverables in **one (1) zip file** on VLE with your name.

It should contain the following deliverables:

1. Report

A report of not more than **3000 words**.

It needs to contain at least the following sections:

Implementation

Describe how you implemented the state representation and successor function for the corresponding applicable actions, the search strategies and your heuristics. For each heuristic, state why you think it is informative to provide good search guidance. Also state whether it is admissible (with adequate explanation or proof as to why).

Describe how you encoded the domain declaratively using PDDL and why. (Include the actual PDDL files in an Appendix. They do not count towards the word count). Describe how you translated each input maze to a PDDL problem file.

Evaluation

Analyse the performance of each search strategy with each heuristic. Compare different metrics, such as the number of states generated, number of states expanded, time taken to generate the solution, and the quality of the solution (the length of the plan). In the comparison also include the same performance metrics for the online generic PDDL-based solver. (Remember that the online solver runs on different hardware than your local algorithms, so a time-based comparison with the online solver is not relevant).

Any citations and references to any literature you may need to quote or refer to should be done using either the AAAI or IEEE reference style.

2. Source Code

Put the source code of both part 1 and part 2 in separate folders in your zip file.

Demo Presentation

You will be allocated a total of 20 minutes to present your work. The first half of this will be a demo of the working project, while the second half will be dedicated to questions about your implementation and design decisions. (You do not need to prepare separate slides for this.)

The date and time slot for the demo presentation of each team will be communicated at a later date. **All team members are expected to attend.**

Distribution of Work

Each team must include a "Distribution of Work" page, at the back of the report. (This does not count towards the word count limit and can be included after the reference list).

In this page put the names of each team member and a very short description (a few bullet points) of the tasks that each person was responsible for. **Each member must sign this page**, indicating that they agree with the reported distribution of work.

Plagiarism

While this is a group project, each group is expected to work on his own.

Plagiarism is taken very seriously and any suspicion of plagiarism, copying, or claims of the work of others to be one's own will be thoroughly investigated and if confirmed, severe action will be taken according to the regulations of the University of Malta.

All group members should sign this form and it must be scanned and included with your submission.

https://www.um.edu.mt/media/um/docs/faculties/ict/mainfacultyofictfiles/PlagiarismForm.pdf

For more details about plagiarism and collusion you can consult the **University Guidelines on Plagiarism and Collusion** at:

https://www.um.edu.mt/registrar/policiesguidelinesforms/policiesguidelines/students/studentconduct/

Use of Generative AI

The use of generative AI tools is only permitted under very strict conditions. While it can be used to inspire and explore ideas, it **should not** be a replacement to the brainstorming, critical thought and the process of formulating the deliverables.

Wherever Generative AI was used, the following must be declared in an Appendix:

1. The AI tool used
2. For which part of the project.
3. How the tool was used, including the **exact prompts** and how the output was used in the brainstorming or discussion process.
4. The **verification** of the AI response that you applied, including making sure that the response is correct and not hallucinated, and **finding and attributing the original sources**.

For more information, refer to the Dept of AI **"Assessment Policy on the Use of Generative AI Tools".**

Students **may be called** for an **oral examination** if there is reasonable suspicion that the work submitted is not truly their own, in accordance with article **38(1)(f) of the University Assessment Regulations (2009). This includes non-adherence to the policy on the use of Generative AI tools.**

Appendix A: Assessment Criteria

Table 2 serves as a guide to indicate what criteria will be used to assess each task.

	Weak	Satisfactory	Excellent
Implementation Part 1 (40%)	The program does not work. The search algorithms were not implemented correctly, or not all of them were implemented. The heuristics and state evaluation function are not implemented correctly.	The program works correctly. The search algorithms work, and heuristic implementations were provided. The right output is generated. Implementation described in the report.	The program works correctly, offering an easy way to try different algorithms and heuristics. The program finds a solution fairly quickly, showing that the implementation is efficient and keeps track of visited states. Implementation described in detail.
Implementation Part 2 (30%)	No PDDL domain model was provided, or a non-working domain model was provided. No or non-working problem files were provided.	A PDDL domain model was provided which supports the objects and actions described. Working problem files were provided for the respective maze files.	Working PDDL domain model and PDDL problem files were provided. An automatic problem translator has been provided which reads the input maze and generates the respective PDDL problem file.
Evaluation (20%)	No or minimal analysis of the performance of the algorithms.	Comparison of each of the performance metrics with adequate visualisation, on all the algorithms and heuristics, including the metrics of one online solver.	Comparison of each of the performance metrics, with adequate visualisation, on all the algorithms and heuristics, including the metrics of one online solver. A discussion explaining these results and why they were observed.
Language and Style (5%)	Language is unclear. It is evident that the report was not properly proof-read prior to submission.	Language is coherent, with minor grammatical or language issues. The document is properly structured. The right technical terminology is used.	Language is clear, flowing and grammatically correct. The document is properly structured. The right technical terminology is used. The document has been proof-read and very few grammatical or language issues are detectable.
References (5%)	References are missing, or inconsistent. No use of any of the	The report includes a number of references and they have been	Claims made in the report are backed up by references. The

	recommended styles have been used.	adequately included with the right referencing convention, using one of the recommended styles. Minor mistakes in references.	references follow a consistent convention, using one of the recommended styles. A good number of references have been provided.

Table 2: Marking Criteria