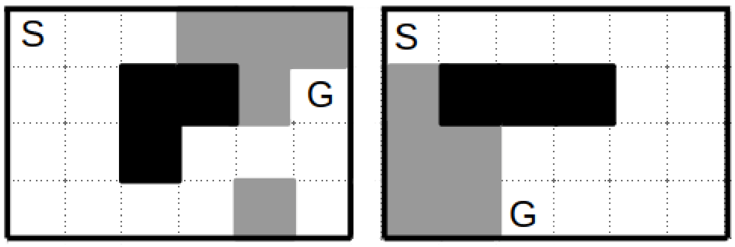
F29AI – Artificial Intelligence and Intelligent Agents

Coursework 1 – Part 2 – A* Search & Automated Planning

You should complete this coursework **individually**. This part of Coursework 1 is worth **20%** of your overall F29AI mark. There are two questions in this part of the coursework: one on A* search (worth **5%**) and one on automated planning (worth **15%**). Details of what to do and hand in, and how you will be assessed, are below.

A* Search - Problem description



The above grids represent two problem environments where an agent is trying to find a path from the start location (S) to the goal location (S). The agent can move to an adjacent square provided the square is white or grey. The agent cannot move to a black square which represents a wall. No diagonal movement is allowed. Moving into a white square has a cost of 1. Moving into a grey square has a cost of 3.

What to do: Undergraduate students, 1-year Edinburgh postgraduate students, and all Dubai postgraduate students

Implement a solution to the grid problem in Java using A* search. Starter code has been provided for you and is available in the accompanying zip file. In particular, the code in the package **uk.ac.hw.macs.search** is a set of classes that can be used to represent a search problem. To implement a specific search problem, you will need to do the following:

1. Define a class that implements the `State` interface. This should include the following:
   a. A method for determining whether a state is a goal state (`isGoal()`)
   b. A method for computing the heuristic value of a state (`getHeuristic()`)
2. Define a class that implements the `SearchOrder` interface. This interface has one public method, `addToFrontier`, that is used to add a set of nodes to the frontier. You can use the costs and/or heuristic values to determine the order that nodes are added to the frontier

The classes in the **uk.ac.hw.macs.search.example** package show examples of these two interfaces being used to implement depth-first search and breadth-first search, as well as a simple integer-based state. The `Main` class in this package shows an example of how to use the classes.

To solve the problem, you will also need to implement the following:
1. An appropriate encoding of the state in the grid by implementing the `State` interface appropriately, including methods for detecting a goal state and computing a heuristic value. You will need to choose or design an appropriate heuristic to use for your search. Include a description of the heuristic in the comments of your code.
2. A class implementing A* search by implementing the `SearchOrder` interface and implementing `addToFrontier` appropriately

Test your code on the two grid problems provided by running A* search on them. Submit the code for the implementation as well as the output. Do not change any of the classes in the **uk.ac.hw.macs.search**: we will test your implementation against the provided classes.

What to do: 2-year Edinburgh postgraduate students only

Describe a solution to the grid problem using A*search. In particular, you should do the following:

1. Draw a graph (with nodes and edges) to represent each of the grid problems as a search problem. Label each of the nodes in your graph and include appropriate costs on the edges.
2. Choose or design an appropriate heuristic to use for both graphs. Describe how your heuristic operates and include the heuristic values on each node in your graph.
3. Apply A* search to each grid and list the states expanded and the goal path for each grid. You do not need to provide a full derivation for each search (unless you wish to do so), however, you should provide at least the first 5 steps of each derivation with calculations for the f values to demonstrate you understand the application of the A* algorithm in your graph.

Submit your answers in a document in PDF format.

Automated Planning - Problem description

The Heriot-Watt SeaPort has decided to launch its next-generation submarine for underwater exploration. Operations on board the submarine will be controlled by mission plans generated using an automated planner that will direct the people, autonomous agents, and submarine's controls. The submarine itself is quite large with multiple sections, including the bridge, launch bay, science lab, sickbay, and possibly others. Different types of personnel serve on board the submarine: the captain, engineers, science officers, security personnel, and navigators. The submarine is also equipped with specialised underwater vehicles (mini-subs) that can perform various tasks. Some of the submarine's operations are described in the following list (which isn't exhaustive):

1. The captain can order the submarine to travel to a given underwater region, provided the captain is on the bridge and a navigator is present to receive the order.
2. The submarine can travel to an underwater region provided a navigator is on the bridge to control the submarine and has received an order from the captain. The navigator must have received an order to travel to that specific underwater region.
3. Personnel can move between different sections of the submarine, which are connected by doors. Personnel may have to pass through different sections to reach their destination and cannot simply move from any location to any other location directly.
4. Each underwater region can be empty or can contain a ridge, an abyssal plain, and/or a vortex.
5. A drilling mini-sub can be deployed to a ridge to retrieve mineral samples. To do so, the mini-sub must be launched from the submarine, approach the ridge and drill for mineral samples, and return to the submarine. No submarine personnel are required on the lander.
6. An exploration mini-sub can be deployed to an abyssal plain to install an underwater sensor net to monitor the plain. An engineer must operate the mini-sub and set up the sensor devices. To complete the mission the mini-sub must land on the plain, the underwater sensor net must be deployed, and the mini-sub and engineer must return to the submarine.
7. An exploration mini-sub can be deployed to an abyssal plain that has an underwater base to check on the people working in the base. The captain must be part of the mission. If the base has been taken over by Atlanteans and the submarine's security personnel aren't present, the captain will be injured. To complete the mission the mini-sub must land at the base, the captain must meet the base's leaders, and then the mini-sub and submarine personnel must return to the submarine.
8. Mini-subs can be launched from the launch bay provided an engineer is in the launch bay to operate the launch controls. Mini-subs can be retrieved back into the launch bay if an engineer is present. When a drilling mini-sub is retrieved, any mineral samples collected are automatically transferred into the launch bay. When an exploration mini-sub is retrieved, all personnel on the mini-sub disembark into the launch bay.
9. Injured crew members can be healed in the submarine's autonomous sickbay.
10. Only science officers can move mineral samples around the submarine. Mineral samples can be studied in the science lab if a science officer is present.
11. A vortex can be studied if the submarine is in an underwater region with a vortex. A science officer can perform a vortex scan from the science lab.
12. The captain can order the navigator to direct the submarine into a vortex.
13. If the submarine tries to enter a vortex without its pressure shields up it will be destroyed. Engineers can activate the pressure shields from the bridge.
14. If the submarine enters a vortex it is immediately transported to another underwater region.
15. When the submarine returns to the SeaPort, reports of all mineral samples and vortex studies, all plains with sensor nets, and all status reports of underwater bases must be communicated back to the SeaPort to successfully end the mission.

At launch, the submarine is given a series of missions that it must complete. These missions might include retrieving mineral samples from ridges, studying vortices, placing sensor nets on abyssal plains, and checking underwater bases. The submarine initially starts at the SeaPort and must return there at the successful completion of its missions.

What to do: all students

1. **PDDL implementation:** You must model the submarine exploration domain in PDDL by defining the properties, objects, and actions that are needed to describe the domain. Note that the planning domain is described at an abstract level and is somewhat incomplete, with certain pieces of information missing. You must make design decisions as to how you will represent the knowledge and actions necessary to encode this scenario as a planning problem. Some requirements are more difficult than others. It is strongly recommended that you try to implement the domain incrementally, ensuring that some parts of the domain work correctly before moving on to others. Use the example domains from the PDDL lectures as a starting point for your solutions. You may also find that the planning time increases as you add more complexity. You may have to consider whether an alternative knowledge representation leads to a better solution. You may use the planning tools available at http://editor.planning.domains/, the Fast Forward (FF) planner, or the Fast Downward planner. You should ensure that your solution works with one of these planners. Note that the performance of certain planners (e.g., FF and Fast Downward) may outperform that of the web-based planner. Make sure you test your solution on a series of different problem scenarios.

2. **Additional feature:** In addition to the domain described above, design an additional feature in PDDL to add to the domain (e.g., new personnel that can perform some task, a new function for a mini-sub, etc.) that isn't included in the above domain description. Add this feature to your domain and test it.

3. **Report:** Write a short report (maximum 2 pages) outlining your solution, design decisions, layout of the submarine/underwater regions, and additional feature. The report will be used to help understand your code.

What to do: 1-year Edinburgh postgraduate students and all Dubai postgraduate students

In addition to the above instructions, postgraduate students in a 1-year programme should also answer the following questions:

- How well does the domain scale? Choose a planning problem and increase some feature of the domain (e.g., the number of locations to explore, the number of people on the submarine, the number of mini-subs, etc.). Does this have an effect on the planning times or the plan length? Does the planner ever fail to generate a plan? Include a brief description of these results in the report (maximum 2 additional pages). Use tables and graphs to illustrate the data you collected. Speculate on the robustness of your domain and how you might improve it to scale to larger problem instances.

What to hand in

- **Java code / A* search solutions:** Submit your Java code for A* search and your particular implementation of the two grid search problems. Alternatively for 2-year postgraduate students, submit your solutions to the A* search problems as a PDF file. Code and reports will be checked for plagiarism.

- **PDDL source files:** Submit your PDDL source files (domain + problems) in a zip/tar file. Make sure your source files have comments describing the properties and actions you've defined. Your solution should consist of a single PDDL domain file and at least 4 different PDDL problem files illustrating different scenarios that are supported by your domain. One of the problem files must test your additional feature. You should aim for comprehensive scenarios that support multiple missions in each problem file. Your source files will be tested to see if they are operational and checked for plagiarism.

- **Report:** Submit your report as a PDF file. Your report will be checked for plagiarism.

Deadlines

The deadline for submitting Coursework 1 – Part 2 is **Thursday, 29 October 2020**. Submissions are due by **3:30pm (Edinburgh local time)** for the Edinburgh Campus, **5:00pm (Dubai local time)** for undergraduate students at the Dubai Campus, and **11:59pm (Dubai local time)** for postgraduate students at the Dubai Campus. Submit your coursework on Vision in the **Coursework** section of F29AI.

Assessment

This coursework will count towards **20%** of your overall course mark for F29AI and will be marked out of **25 marks**. You will be assessed on the quality and content of your report, the style of your PDDL code, and the correctness of your solution.

A* Search: Java code (5 marks)

Your Java code will primarily be marked on its correctness with respect to the two grid problems. We will also check the description of the heuristic you are using in your solution.

0 **None**	**1** **Poor**	**2** **Fair**	**3** **Good**	**4** **Very good**	**5** **Excellent**
No code submitted.	Major problems with code. Solution is incorrect and/or code does not run.	Code partially works but there are major problems with code structure, solution, and/or heuristic.	Good code and solution. Code runs almost perfectly but there are problems with code structure, solution or description of heuristic.	Very good code and solution. Code runs perfectly. Small problems with code structure, solution, or description of heuristic.	Exceptional code and solution. Code runs perfectly. Heuristic is well defined.

A* Search: report (5 marks)

Your report will primarily be marked on its correctness with respect to the two grid problems. We will also check the description of the heuristic you are using in your solution.

0 **None**	**1** **Poor**	**2** **Fair**	**3** **Good**	**4** **Very good**	**5** **Excellent**
No report submitted.	Major problems with report. Solution is incorrect and/or many parts of the solution are wrong or missing.	Major problems in the report. Some parts of the solution partially incorrect or missing. Description of the solution could be improved in many places.	Good report. Solution is mainly correct but there are some minor problems. Description of the solution could be improved.	Very good report. All parts of the solution are described and the solution is correct but there are some small problems with the description that could be improved.	Exceptional report. All parts of the solution are well described and solution is perfect.

Automated Planning: Quality of report and solution (5 marks)

The quality mark will primarily be based on your 2-page written report which will outline your solution, design decisions, and additional domain feature. Your report should be well prepared and presented, with good English usage and grammar, and appropriate visual appearance. Your PDDL code will also be checked and the report will be used to help supplement the code in understanding your overall approach to the problem and the quality of your solution. Usual program quality criteria (e.g., use of whitespace, comments, naming conventions, etc.) will apply here to assess the readability of the code. The marker will be looking to see if you understand how to write PDDL domains and problems and have made good use of the language features that are available.

Postgraduate students only: Reports by postgraduate students may contain an additional 2 pages and should also include details about your scalability/robustness study (including appropriate use of tables and graphs), and appropriate conclusions. Questions concerning planning time/length, robustness, and scalability improvements should be addressed in the report.

0 **None**	**1** **Poor**	**2** **Fair**	**3** **Good**	**4** **Very good**	**5** **Excellent**
No report submitted.	Weak preparation and unstructured report. Weak quality solution. Basic domain features. PDDL code is not understandable. [Postgraduate students: Questions not answered or addressed.]	Adequate preparation and structure of report. Fair quality solution. Mostly basic domain features. PDDL code is very difficult to understand. [Postgraduate students: Basic information provided with few conclusions.]	Thorough preparation and structure of report. A good quality solution overall with a mix of basic and advanced domain features. PDDL code is mainly well written, clear, and understandable. [Postgraduate students: Good description with all required details and appropriate conclusions.]	Very thorough preparation and structure of report, with many design details included. A good mix of domain features, with advanced features enhancing the overall approach. PDDL code is very well written, completely clear and understandable. [Postgraduate students: Very good description with all required details & strong conclusions.]	Exceptional preparation and structure of report. Excellent design designs and use of advanced features which enhance the overall approach. PDDL code is excellent, with all aspects of the code easily understandable. [Postgraduate students: Excellent content, data, and conclusions.]

Automated Planning: Correctness of solution (15 marks)

The correctness mark will primarily be based on the PDDL code you have supplied and how correctly it implements the coursework specification. You may submit a single PDDL domain file and multiple PDDL problem files in a single zip/tar file. The supplied files should provide coverage of the various scenario requirements and demonstrate correct behaviour. The marker will test a selection of the specified PDDL files for plan correctness. If you have developed your solution using a planner not mentioned in class, this should be clearly indicated in your report; however, the PDDL files will only be tested on editor.planning.domains, FF, or Fast Downward. Correctness will be assessed not only against the coursework requirements but also with respect to the specific implemented solution. (I.e., non-obvious or incorrect/missing action preconditions or effects may lead to strange plan output and mark deductions.) Up to 2 marks will be allocated for the correctness of the additional feature. The additional feature should be similar in terms of complexity to the requirements in the main specification (and may be more complex if you'd like), and you are encouraged to be creative in your solutions. Trivial additional features will receive very few marks.

Postgraduate students only: The scalability study demonstrated in the report will be assessed as part of the correctness mark, with 2 marks (of the total 15 marks allocated for correctness) reserved for the correctness of the scalability study.

0 **None**	**1-3** **Poor**	**4-6** **Fair**	**7-9** **Good**	**10-12** **Very good**	**13-15** **Excellent**
No source code submitted.	Weak solution. Many important requirements or test cases missing and/or not working perfectly.	Adequate solution. Some requirements implemented but important requirements missing. Test cases do not provide complete coverage. Some test cases may not work perfectly.	Thorough solution. Majority of requirements are met. Choice of test cases is convincing. Very few test cases do not work perfectly.	Very thorough solution. All requirements met and choice of test cases provides maximum coverage of requirements. Solution plans are quite convincing. Everything works perfectly.	Exceptional solution. All requirements are easily met and well demonstrated. Test cases are comprehensive and demonstrate robust behaviour. Everything works perfectly.

Additional notes

This is an **individual coursework** assignment. The Java/PDDL code and reports will be checked for plagiarism. You are responsible for ensuring that your submitted files are readable. The marker should not need to make any changes to submitted code files to get them to work. If the files are unreadable or do not run, you will receive **0 marks** on that part of the assessment.

Feedback

Individual written feedback will be provided to students approximately three working weeks after the submission of Coursework 1.

Learning Objectives

This coursework is meant to contribute to the following high-level aims for F29AI:

- To introduce the fundamental concepts and techniques of AI, including planning, search, and knowledge representation.
- To introduce the scope, subfields and applications of AI, including autonomous agents.
- To develop skills in AI programming in an appropriate language.

It is also meant to contribute to the following specific learning objectives for the course:

- Critical understanding of traditional AI problem solving and knowledge representation methods.
- Use of knowledge representation techniques (such as predicate logic).
- Critical understanding of different systematic and heuristic search techniques.
- Practice in expressing problems in terms of state-space search.
- Broad knowledge and understanding of the subfields and applications of AI.
- Detailed knowledge of one subfield of AI (e.g., planning) and ability to apply its formalisms and representations to small problems.
- Detailed understanding of different approaches to autonomous agent and robot architectures, and the ability to critically evaluate their advantages and disadvantages in different contexts.
- Practice in the implementation of simple AI systems using a suitable language.
- Identification, representation and solution of problems.
- Research skills and report writing.
- Practice in the use of information and communication technology (ICT), numeracy, and presentation skills.

Late submission of coursework

Coursework deadlines are fixed and individual coursework extensions will not be granted. Penalties for the late submission of coursework follow the university's policy on late submissions:

- The mark for coursework submitted late, but within 5 working days of the coursework deadline, will be reduced by 30%.
- Coursework submitted more than 5 working days after the deadline will not be marked.
- In a case where a student submits coursework up to 5 working days late, and the student has valid mitigating circumstances, the Mitigating Circumstances policy will apply. Students should submit a Mitigating Circumstances application for consideration by the Mitigating Circumstances Committee.

The MACS School policy on coursework submission is that the **deadline for coursework submissions**, whether hard-copy or online, is **3:30pm (Edinburgh local time)** for the Edinburgh Campus and **5:00pm (Dubai local time)** for the Dubai Campus. The University Policy on the Submission of Coursework can be found here: https://www.hw.ac.uk/services/docs/learning-teaching/policies/submissionofcoursework-policy.pdf

Mitigating Circumstances (MC)

There are circumstances which, through no fault of your own, may have affected your performance in an assessment (exams or other assessment), meaning that the assessment has not accurately measured your ability. These circumstances are described as **mitigating circumstances**. You can submit an application to have mitigating circumstances taken into account. Full details on the university's policies on mitigating circumstances and how to submit an application can be found here:
https://www.hw.ac.uk/students/studies/examinations/mitigating-circumstances.htm

Plagiarism

"**Plagiarism** is the act of taking the ideas, writings or inventions of another person and using these as if they were your own, whether intentionally or not. Plagiarism occurs where there is no acknowledgement that the writings, or ideas, belong to or have come from another source." (Heriot-Watt University Plagiarism Policy). This coursework must be completed independently:

- Coursework reports must be written in a student's own words and any submitted code (e.g., PDDL) in the coursework must be your own code. Short sections of text or code taken from approved sources like the lecture examples may be included in the coursework provided these sources are **properly referenced**.
- Failure to reference work that has been obtained from other sources or to copy the words and/or code of another student is plagiarism and, if detected, this will be reported to the School's Discipline Committee. If a student is found guilty of plagiarism, the penalty could involve voiding the course.
- Students must **never** give hard or soft copies of their coursework reports or code to another student. Students must **always refuse** any request from another student for a copy of their report and/or code.
- Sharing a coursework report and/or code with another student is **collusion**, and if detected, this will be reported to the School's Discipline Committee. If found guilty of collusion, the penalty could involve voiding the course.

Plagiarism will be treated extremely seriously as an act of academic misconduct which will result in appropriate student discipline. All students should familiarise themselves with the university policies around plagiarism which can be found here: https://www.hw.ac.uk/students/studies/examinations/plagiarism.htm