F29AI - Artificial Intelligence and Intelligent Agents

Coursework 1

Search and Automated Planning

You should complete this coursework **working in pairs**. Coursework 1 has **two parts** (Search algorithms and automated planning using PDDL) and is worth **25%** of your overall F29AI mark. Details of what you should do and hand in, and how you will be assessed, are described below and on Canvas.

Part 1: Solving and Analyzing Sudoku with Search Algorithms

Part 1A (5 marks)

Artificial Intelligence (AI) is incessantly getting more sophisticated and pervasive with every passing day, if not minute. While it is often deemed a staple for science fiction fans, its applications in day-to-day life are tremendous. You probably utilize AI and its faction, Machine Learning, in more places than you can probably imagine.

Sudoku is more than just a number puzzle, it is a rich problem space that exemplifies several core concepts in Artificial Intelligence (AI). It consists of a 9x9 grid, and the objective is to fill the grid with digits in such a way that each row, each column, and each of the 9 principal 3x3 sub-squares contains all of the digits from 1 to 9. You need to propose and develop an approach to an intelligent sudoku solver.

- Define Sudoku formally as a constraint satisfaction problem.
- What are the variables, domains, and constraints?
- Discuss the time complexity of brute-force search vs. backtracking in Sudoku.

Part 1B (15 marks)

Your task: Design and implement an intelligent system in Python or Java that can efficiently solve any valid 9×9 Sudoku puzzle. Your report must leverage concepts from Artificial Intelligence, particularly search algorithms and constraint satisfaction problem (CSP) techniques. The purpose of this assignment is to help you understand how AI can be applied to structured problems by comparing the effectiveness and efficiency of different problem-solving strategies.

Project Requirements: Your task is to build an intelligent Sudoku Solver. There are many possible ways to approach this problem, such as: a brute-force search, a constraint satisfaction

approach (e.g., backtracking with pruning, forward checking, or arc consistency), or a machine learning–based method.

Choose ONE method to implement a Sudoku solver. Once you have built your solver, you will critically analyse and compare your approach with an A* search–based Sudoku solver in theory. Please note: You are NOT required to implement an A*search algorithm. Instead, you must provide a theoretical analysis that compares your implemented solver with an A*search algorithm approach to solving the sudoku problem.

Input: The program should accept input puzzles from .txt or .csv files. Each file will contain a 9x9 Sudoku grid using the standard format: Empty cells can be represented using 0 or a blank character.

Some websites to look for Sudoku puzzles:

- https://www.websudoku.com/
- https://sudoku.com/

Output: The program must display the solved Sudoku puzzle in a clean and readable format. For example, it could also print other relevant metrics based on your approach used, e.g. Total number of steps, recursive calls, or backtracks, Total execution time (in milliseconds or seconds). You are advised to test your solutions with different Sudoku puzzles.

Optional : Students who implement a Graphical User Interface (GUI) that allows users to input puzzles, run the solver, and visualize the solving process through a nice GUI will be awarded some bonus marks. See the rubrics for more details.

Report (See details below what to hand in): Your report should clearly describe the implementation of your chosen method (supported by screenshots and/or code snippets). You must provide a thorough theoretical analysis comparing your implemented solver with how an A*search algorithm would approach the same problem. You should describe how you tested your solution, e.g. with different input.

Part 2: Automated Planning

The European Space Agency (ESA) is preparing the *European Large Logistics Lander* (EL3) mission[1], a robotic lunar exploration initiative designed to explore and retrieve samples from the Moon. The mission involves deploying **Landers**, each carrying a **Rover**.

Given the harsh and unpredictable conditions on the Moon, all operations must be fully autonomous. You have been tasked with developing a planning solution for the deployment and task execution of the Rover, which will be released from the Lander at the designated site.

Once deployed, the Rover will perform a series of scientific and exploratory tasks:

- Navigate through a series of lunar waypoints.
- Capture high-resolution images of geological features.
- Perform subsurface scans using ground-penetrating radar.
- Collect soil and rock samples.

[1]See this video from ESA's YouTube Channel for an illustration `https://www.youtube.com/watch?v=75MQSSJHIeY`

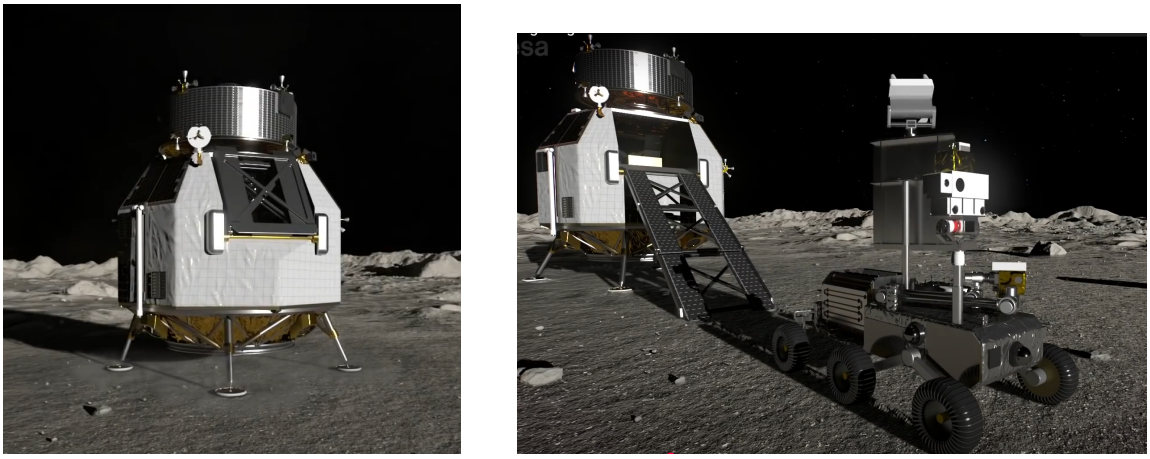

Figure 1: Pictures of a lander (left) and a rover being deployed (right).

Data from images and scans can be transmitted wirelessly back to the lander. Physical samples, however, must be returned manually by the rover at the end of its mission.

A decision has been made to use PDDL and an off-the-shelf planner to model the mission planning problems and define the lunar environment in which the rovers will operate. The PDDL domain must represent the various types of entities involved in the mission, including landers, rovers, and lunar surface locations.

The domain should define predicates that describe the world state, such as:

- The current position of each rover.
- Connectivity between surface locations.
- Whether a rover has captured or transmitted data (e.g. images, scans).
- Whether a rover has collected physical samples.
- The association between rovers and their respective landers.

Each problem instance will define:

- The initial state of the mission (e.g. lander landing sites, rover deployment status, terrain layout).
- The goal state, based on mission objectives (e.g. specific samples to be collected, data to be transmitted, rovers to return to landers).

Part 2A & 2B (15 Marks)

Part 2A: Modelling the Domain

The first step is to understand the **domain**. For this part of the coursework, you need to implement the domain of the planning problem.

Describing the World State

Define the types of objects involved in the mission and the predicates representing the system's state. These predicates should indicate whether the Rover is at a specific location, has collected an image, the connection between two locations, and other relevant states.

Defining Actions

Define the actions that can be performed. These actions may include deploying the Rover, moving between locations, taking pictures at specific locations, using radar to perform subsurface scans, and transmitting data back to the Lander. Each action has specific preconditions that must be met for execution. For example, the Rover must be at the correct location to take a picture, or it must have collected the data before it can transmit it to the Lander. Each action also has certain effects that will modify the overall world state if the preconditions are met. For instance, if there is a path between two locations and the Rover is at one of them, it can move along the path, resulting in it no longer being at the starting point but at the new location.

RESTRICTIONS

- The Lander is capable of landing at any location but remains stationary once landed and cannot relocate.
- The Rover has limited memory, capable of storing only one piece of data at a time; both "image" and "scan" are data.
- To collect a sample, the Rover must pick it up from a location and return to the Lander to store it.
- Each Lander can store only one sample.

Part 2B: Modelling the Problems

For this part, you'll create problem files for **Mission 1** and **Mission 2**. Each problem should define the **initial state** and **goal** so the planner can generate valid plans.

Mission 1

- The Rover begins its journey from the Lander (i.e. undeployed).
- Use the map in Figure 2 to configure connections and paths between waypoints.

Mission Goals:

- Save an image at waypoint 5.
- Save a scan at waypoint 3.
- Collect a sample from waypoint 1.

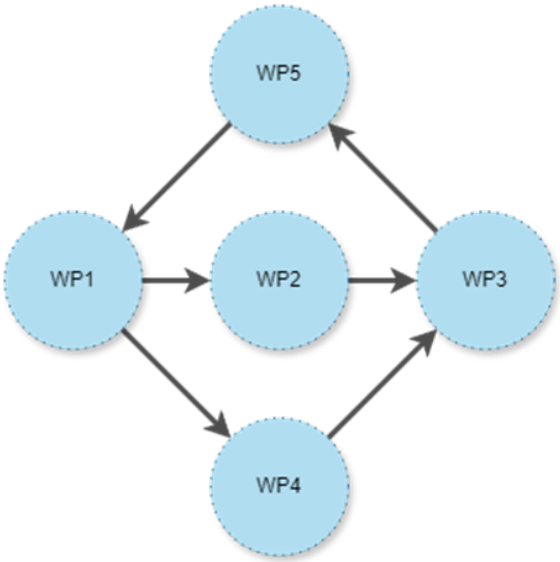

Figure 2: Map for Mission 1.

Mission 2

- Use the map in Figure 3 to configure connections and paths between waypoints.
- Use two Landers for this problem.
- The first Rover starts already deployed at waypoint 2, and its Lander is also at waypoint 2.
- The second Rover starts undeployed.

Mission Goals:

- Save an image at waypoint 3.
- Save a scan at waypoint 4.
- Save an image at waypoint 2.
- Save a scan at waypoint 6.
- Collect a sample from waypoint 5.
- Collect a sample from waypoint 1.

Part 2C: Extension (5 marks)

In this section, you will **extend the domain** by introducing new features and create a problem file based on the updated domain and the mission below.

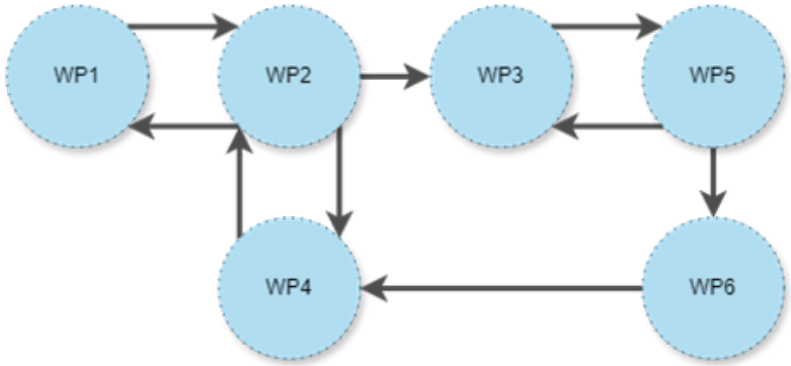

Figure 3: Map for Mission 2.

Mission 3

Use the same map (Figure 3) and initial setup as in Mission 2 above. The mission goals remain unchanged. In addition, consider the following extensions:

- You have two brave astronauts, Alice and Bob, to assist the mission. Alice is stationed in Lander 1, and Bob in Lander 2.
- Each lander includes two internal areas: the control room and the docking bay. Astronauts can move between these areas.
- A Rover can only be deployed or a sample retrieved when an astronaut is present in the docking bay.
- Data transmission from the Rover to the Lander is only possible when an astronaut is in the control room.
- Initially, Alice is in the docking bay, while Bob is in the control room.

Instructions

1. **Download the Starter Files:** Download the **PDDL.zip** file containing the starter PDDL code folders and files.
2. **Folder Structure:** The zip file contains two subfolders:
   - **PartA&B:** For Part 2A and Part 2B.
   - **PartC:** For Part 2C.
3. **Getting Started:** Begin by modifying the provided domain and problem files. The domain and problem names are already set up for you.

PDDL Implementation

Do not use any features of PDDL that we have not covered in the course. Include comments in your PDDL files to describe important sections of your code. **You do not need PDDL features that we have not covered in the course.** You should clearly indicate in your report what planner you have used to test your solution. 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.) 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.

We recommend using Visual Studio Code (`https://code.visualstudio.com/`) as code editor for PDDL, and the PDDL support extension: `https://marketplace.visualstudio.com/items?itemName=jan-dolejsi.pddl`.

Once the extension is installed you can run the planner by right clicking in the domain or problem file and click on "PDDL: Run the planner and display the plan"

You can select the online plan-as-service operation mode offered by planning.domains, with one of this Best First Search planners: "BFWS –FF-parser version", "LAMA-first"

IMPORTANT: if you have multiple domain files make sure the name of your domain matches the one in the problem you want to run the planner with as sometimes there could be errors.

Example:

Domain:

```
(define (domain lunar)
...
)
```

Problem:

```
(define (problem lunar-mission-1)
    (:domain lunar)
...
)
```

Alternatively, if you are having problem setting up the development environment you can use the online editor: `https://editor.planning.domains/`. If you are using Visual Studio Code and you want to run planners offline you can always download one and use it on your local machine, here are some of the most important classical planners:

- Fast Downward (`https://www.fast-downward.org/HomePage`)
- FF (`https://fai.cs.uni-saarland.de/hoffmann/ff.html`) (look for FF-v2.3)

- Pyperplan (`https://github.com/aibasel/pyperplan`)

What to hand in

Submit a single zipped folder (**accepted formats: .zip or .rar**) that includes the following:

1: Report

A report detailing Part 1A and Part 1B search showing a clear step-by-step approach to solving the problem. The report should contain screenshots of your artefact, code snippets implementation as well as the output and justification of your approach using appropriate references where necessary (Part 1A and 1B maximum 4 pages). Compare your approach to an A* star algorithm for solving a sudoku. You may use a comparison table, show the time taken to solve the puzzle, show number of backtracks, look at Success/failure cases and you could include charts/graphs if needed. You can discuss which sudoku solver techniques work best and why?

The same report should include Part 2: PDDL (maximum 3 pages), in which you briefly discuss how you structured your domain, for example, the types of objects, predicates, and actions you included. You should also list the problem instances included in your code, along with the planner used to test your domain and problems. This section of the report will help the markers better understand your code.

At the beginning of your report, clearly state the group you have signed up for on Canvas (e.g., Edinburgh CW 1 42), along with the names and student ID numbers of all group members. At the end of your report, clearly provide the link to your video demo.

2: Code (Java/Python Sudoku files and PDDL files)

Submit the code for the implementation of the Sudoku solver problem in Part 1B. Make sure your code includes appropriate comments in the parts of the code you implemented.

Submit your PDDL source files consisting of those included in the starter package **PDDL.zip**. Make sure your source files have comments describing the properties and actions you've defined. Your source files will be checked for plagiarism and tested to see if they are operational.

3: Video Demo (link)

A video walk-through of the implementation (both students need to be present in the recorded video) explaining the code implementation from a suitable Java/Python IDE and design decision made for the Sudoku solver. In the same video, you should also include a demo of the PDDL domain, predicates, output etc by opening the planner, generating the plans from the given tasks. The video should not be more than 8 minutes long.

Please prepare the video recording and submit the **link** to us. One convenient way is to use Microsoft Stream or Teams to record your video (note: you may use other tools as you wish) and then provide the Sharepoint / OneDrive / Google Drive link in your report. It is your responsibility to ensure the link is accessible, otherwise no marks will be awarded to this part.

Deadlines

The deadline for submitting Coursework 1 (all parts) is **Thursday, 23 October 2025**. Submissions are due by **15:30 (Edinburgh time)** for the Edinburgh Campus, **23:59 (Dubai time)** for the Dubai Campus, and **23:59 (Malaysia time)** for the Malaysia Campus. Details on how to submit your coursework will be posted on Canvas.

Feedback

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

Additional notes

This is a group coursework assignment. All submitted files will be checked for plagiarism. You must conform to the naming conventions described in this document. If files are unreadable or code does not run or video does not play, you will receive 0 marks on that part of the assessment. You are strongly encouraged to read the Student Guidance on Use of GenAI in Learning and Teaching . Whilst you can use it for idea generation the work submitted must be your own.

Assessment

This coursework will count towards 25% of your overall mark for F29AI and will be marked out of 45 marks.

Part 1A (5 marks).

0 (None)	1 (Poor)	2 (Fair)	3 (Good)	4 (Very Good)	5 (Excellent)
No understanding of variables, domains, constraints or complexity.	Very minimal definition, vague or incorrect. Partial mention of components with errors or confusion.	Correctly identifies variables/domains or constraints but not all. References could have been improved.	Good solution.Correct CSP formulation but no complexity analysis. Lack of references.	Correct formulation and brief complexity comparison with an appropriate number of references.	Full correct CSP model and accurate, clear complexity discussion (brute-force vs. backtracking) with a good number and reliable references.

Part 1B Sudoku Solver (15 marks)

Part 1B (15 marks)

0-2	3-5 (Poor)	6-8 (Fair)	9-11 (Good)	12-14 (Very Good)	15 out of 15 (Excellent)
No code submitted. No video demo or very poor demonstration.	Major problems with code. Solution is incorrect and/or code does not run. Demonstration shows that the artefact is poor, lacks functionality or does not work, and falls well below expectation.	Code partially works but there are major problems with code structure, solution. Demonstration shows that the artefact has limited functionality. Fair report.	Good code and solution. Code runs almost perfectly but there are problems with code structure, solution. Demonstration shows that the artefact is equal to expectation. Good report with good comparisons.	Very good code and solution. Code runs perfectly. Small problems with code structure, solution. Very good demonstration. Very good report with good comparisons. Nice GUI implemented.	Exceptional code and solution. Code runs perfectly. Exceptional demonstration that proves the artefact goes far beyond expectation. Excellent report with excellent comparisons and references. Exceptional GUI implemented.

Part 2A & 2B (15 marks)

0–2 (None)	**3–5 (Poor)**	**6–8 (Fair)**	**9–11 (Good)**	**12–14 (Very Good)**	**15 (Excellent)**
No submission, entirely irrelevant content, or incoherent attempt. May include placeholder text or vague ideas with no structure. Demonstrates no meaningful understanding of domain or problem modelling. No working implementation or demonstration.	Submission contains major errors or omissions. Object types, predicates, actions, initial states, or goals are mostly incorrect or missing. Very limited understanding of domain and problem modelling. Implementation is incomplete or fails to run. Demo is missing or unclear.	Some components are present but with significant issues. Object types, predicates, actions, initial states, or goals are partially incorrect or incomplete. Problem logic is weak or inconsistent. Implementation runs with errors or limited functionality. Demo shows partial execution.	Most components are correctly defined. Minor issues in predicate/action definitions, initial state, goal specification, or waypoint configuration. Domain and problem logic are generally sound. Implementation runs correctly with minor issues. Demo clearly shows functionality.	All required components are present and mostly correct. Initial states and goals are well defined. Waypoint connections and domain elements are mostly accurate. Minor improvements could enhance clarity or completeness. Implementation is functional and well-structured. Demo is clear and informative.	All components are clearly and correctly defined. Object types, predicates, actions, initial states, goals, and waypoint configurations are precise, complete, and logically consistent. Domain and problem are modelled with excellent clarity and insight. Implementation is robust and fully functional. Demo is polished, well-structured, and effectively communicates the solution.

Part 2C (5 marks)

0 (None)	1 (Poor)	2 (Fair)	3 (Good)	4 (Very Good)	5 (Excellent)
No submission or entirely irrelevant content. No demonstration of domain extension.	Submission contains major errors or omissions. The domain extension is mostly incorrect or missing. Astronaut-related features are poorly integrated or not reflected in the problem file. Demonstration is unclear or missing.	Some attempt to extend the domain is made, but with significant issues. Astronaut roles or constraints are partially implemented or inconsistent. Problem file lacks coherence. Demonstration shows limited functionality or understanding.	Most extensions are correctly implemented. Astronaut features are generally well integrated, with minor issues in logic or completeness. Problem file is mostly valid. Demonstration is clear and shows working implementation.	All required extensions are present and mostly correct. Astronaut roles and constraints are clearly modelled. Problem file is coherent and aligns well with the mission goals. Demonstration is well-structured and effectively communicates the new features.	All components are clearly and correctly defined. Domain extensions are precise, logically consistent, and well integrated. Problem file reflects excellent understanding and implementation of the new features. Demonstration is polished, insightful, and clearly highlights the astronaut-related enhancements.

Overall Presentation of work and Communication (5 marks)

0 (None)	1 (Poor)	2 (Fair)	3 (Good)	4 (Very Good)	5 (Excellent)
Elements of disorganisation/poor presentation/poor communication or expression or Communications too brief or rambling, inappropriate to context or purpose, with many errors/omissions, inadequately expressed/presented. Little or no clear introduction to the topic.	Organisation and presentation of work and communications adequate in most contexts, with some mistakes/irrelevancies. Outlines a basic introduction to the topic with limited resources from the literature.	Satisfactory organisation and presentation of work, communications mostly appropriate to the context/purpose. Identifies relevant key themes using appropriately cited sources.	Presentation and organisation of work appropriate to context and purpose, communication clear. Detailed overview of the topic supported by a range of appropriate sources.	Excellent presentation and organisation of work and fluent communication in most contexts. Good overall report supported by a wide range of literature from good resources.	Exceptional presentation and organisation of work and fluent communication in all contexts. Evidence of independence of thought and ability to build upon experience, supported by literature from high quality resources.

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 15:30 (Edinburgh time) for the Edinburgh Campus, 23:59 (Dubai time) for the Dubai Campus, and 17:00 (Malaysia time) for the Malaysia Campus. The Submission of Coursework policy 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`