Session	2025/2026	**Semester**	Autumn
Module Name	Symbolic Artificial Intelligence	**Code**	COMP3070
Module Convenor	Doreen Sim Ying Ying		

Coursework Name	Coursework 1 **Weight**: 20%
Coursework Overview	Assessment timetabling usually involves allocating exams into different timeslots while satisfying multiple constraints. This coursework aims to assign an ideal examination time and place for each student, assuming the scheduler knows in advanced which examinations they must take. In this problem, there is a set of students S, a set of exams E, an ordered set of time slots T, and a set of rooms R. Supplied also the set of pairs (s, e), where $s \in S$ and $e \in E$, defining the relation between students and exams. It is also made available to the scheduler the information about capacity $c(r)$ of each room $r \in R$. The following **constraints** should be respected and included when preparing the exam timetable. Set **FIVE main constraints and at least TWO additional constraints** for your solution(s). 1. Each exam must be timetabled in exactly one room and exactly one slot. 2. There can be, at most, one exam timetabled in a room within a specific slot. 3. The number of students taking an exam cannot exceed the capacity of the room where the exam takes place. For example, if three students need to take exam $e \in E$ and the room $r \in R$ has capacity $c(r) = 2$, then e cannot take place in r. 4. A student cannot take exams in consecutive time slots. For example, if the slots are $T = \{t_1, t_2, t_3\}$ and student s took an exam in slot t_1, then student s is not allowed to take another exam in slot t_2. 5. It can be room capacity constraint, exam duration constraint, invigilator assignment constraint, number of student against number of exam constraint or the like. Above constraints are just suggestions as students can always think of many other constraints.
Coursework Details	Hence, the coursework objective is to implement a Z3-based solver for Assessment Timetabling Problems. Please note the difference between the terminology "problem" and the "problem instance" (or simply an instance). A "problem" is a generic description of a family of mathematical questions (in our case, the Exam Timetabling Problem). In contrast, the "problem instance" is a mathematical question with specific parameters, constraints, and etc. The proposed solver should do the following task in the following order. 1. Read the **problem instance** from a `.txt` file. A set of unique problem instances will be provided for development purposes.

	2. Checks the satisfiability of the problem; whether there is an exam timetable that satisfies all the constraints. a) If the problem instance is unsatisable. o Print `unsat'. b) If the problem instance is satisable: o Print `sat'. o Print out the obtained exam timetable 3. (**deeper analyses**) Measure the time taken by Step 2 and then print it in milliseconds. When completed, think of **anymore novelty and/or comparisons of your answers** to earn possible extra mark(s). 4. **Analyses and comparisons** of **all alternative solution(s)** with the **main solution**. **Discussions** on each of the solution strengths and weaknesses. Within reason, you can use any Z3 interfaces to implement your solver (e.g., C++, Java, Python, C, Julia, etc). However, only a **Python template** is prepared for your convenience. Several input files with satisfiable and unsatisfiable instances are provided. The python template includes a function to read the input data.
Deliverable (a brief description of what is to be handed in; e.g. 'software', 'report')	▪ A report. ▪ The solver's code in Python code
Format	▪ The report should be submitted in `.pdf` format. ▪ The solver code in a respected programming language file extension. For example, to submit a `.py` file if the solver was implemented in Python.

Issue Date	**15 October 2025 (Wednesday)**
Submission Date and Time	Report and Python code file: **14 November 2025 (Friday)** at 6:00pm
Submission Mechanism	Both the report and code should be submitted as `.pdf` files via Moodle. For the code, please try your best to wrap **all the functionalities** in a file. For example, if you are using Python to complete the coursework, you can store all the functions/classes within a single `.py` file. If this is not possible, kindly zi p all the solver file extensions when attaching it in Moodle.
Late Policy	A late penalty of 5% per calendar day (excluding weekends) will be applied.
Feedback Date (default of 21 days of the published submission date will apply, if blank)	Report and solver code: The feedback will be provided in due course.

Instructions	**Coursework 1** is explained in the general coursework guide uploaded to Moodle . A FAQ will be maintained in Moodle. If you have questions, talk to the module convenor; answers to new questions will be added to the FAQ.
Assessment Criteria	**Part 1: Completeness and correctness of the solution method(s) related to the Formulations/Alternative Formulations (full-fledged) of Problem(s)(30%)** ▪ The proposed solution should terminate with a solution when one exists. ▪ The proposed should behave according to the predetermined goals and specifications. Set **FIVE main and at least TWO added constraints**.

.

- The proposed Z3-based solver for exam timetabling will be evaluated against unseen instances (i.e., instances other than the example provided).

Part 2: Quality and elegance of the solution methods and Python code as well as thorough and correct comments/explanations on the code (20%)

- The code quality encompasses code clarity, maintainability, documentation, refactoring, well-tested, and efficiency.
- The course also challenges the student to develop a compact, elegant mathematical description for checking the satisfiability of the problem.

Part 3 Report (50%)

- The report should consist of the following elements and in the following order

a) Report Contents:
You may consider introducing the solver and language that you use for this coursework, whether you know of any functionality that does not (always) work, etc. While this is **mandatory**, students should **discuss and analyze different alternative solution(s) and main solution, formulations and alternative formulations,** timetabling solving approaches, **all constraint types** being set, elaborate **novelties** and/or give **deeper analyses and comparisons**.

b) Formulations of the problem(s):

It is **mandatory** to consider **all the Formulations and Alternative Formulations of Problems** for your solutions shown in your Python.

While the mathematical formulation is usually more abstract than the actual implementation, it must be detailed enough to explain how it can be implemented with Z3.

Should there is an introduction of new notations, be clear about them.

The **formulations** can be split into several parts: i. e. the part that is unrelated to the constraints and then the formulation for **each constraint type**. Unless obvious, explain why each formulation is correct for **each constraint type** and the selected API functions.

c) Alternative formulation(s) and solution(s):
Refer to section Assessment Criteria part 3 (b)

d) Implementation:

This section can explain the steps taken to compile/execute the solver. Also, explain the logic of the work of your solver; specifically, how does it establish if there exist multiple solutions to the problem? Discuss any known bugs. If the implementation is straightforward, this section will likely be concise.

	e) Evaluation: For each **sample instance**, please report if the solver tackles it correctly. To be more unique, the **analysis** such as how long it takes to evaluate each sample instance, is recommended. Should there is more performance analysis; then it should be reported in this section. **f) Discussions** A discussion of the question in light of the experimental results. **g) Conclusion** This is the report section for giving a clear summary of the main points of the work, reflecting on its successes and limitations, and briefly discussing how to extend the current works, approaches or evaluations if more time is allocated to complete the coursework. **h) References** List of references used in the report. While this is optional, it is recommended to apply the APA style of reference citation. See the APA documentation for more details https://apastyle.apa.org/ ▪ The report will be evaluated according to its **correctness and quality**. ▪ The student is encouraged to produce and discuss **more than one solution approach** (i.e. alternative solution(s) or the like) to score extra marks. ▪ There is no minimum number of words stipulated. **Oral Interview (0%)** ▪ The interview does not affect the mark; however, it is a compulsory part of the coursework. ▪ Failing to attend the interview will result in a mark of **ZERO** (0) for the coursework. ▪ The module convener will set a time slot for the interview, and you may pick the preferred time at your convenience. More detail about the slot opening will be issued around the submission deadline. ▪ The interview will take around 5~10 minutes and will be face-to-face. ▪ During the interview, the agenda involved a demonstration of the solver, testing the solver against unseen instances, followed by Q&A related to the code and report.
Grading Guidelines	**Band: 90-100** Marks in this range are reserved for a superb all-around performance. Work done in all aspects of the project go beyond even high

	expectations. The student has shown a thorough understanding of the problem. All expected tasks have been successfully completed, the project shows depth and engagement with research ideas, and everything has been completed to a high standard. The report could form the basis of a publishable conference/workshop paper. **Band: 80-89** Excellent contributions to all areas of the project. The program and the report exceed expectations; for example, extra functionality is implemented, interesting analysis is performed, et cetera. Exceeded expectations in some areas. Demonstrates knowledge and understanding of the project beyond standard resources covered in the module. A clear appreciation of the project as a whole, its adequacies, limitations and possibilities for future development. The project demonstrates insight and depth beyond that usually expected in undergraduate work. **Band: 70-79** Very good contributions to all areas of the project. All the requirements are met: the solver performs as expected, and the report adequately reflects the work. Able to reflect accurately on the adequacy and limitations of the project's achievements. **Band: 60-69** Good appreciation of background. A good attempt at applying this to the task, with demonstrated ability to cope with difficulties. Good technical skills in several areas. While most of the requirements are met, the solver mainly performs as expected, and the report adequately reflects the work. It might come a little short in some areas—good reflective understanding of the project. **Band: 50-59** The core task is met, albeit with minor weaknesses: satisfactory background reading and a competent attempt at their tasks. Reasonable technical competence demonstrated. Able to reflect satisfactorily on the project. **Band: 40-49** **Pass level.** Competent background reading and appreciation of the project area. Basic technological competence. Some core areas are met, but a decent attempt has been made at them, albeit with significant weaknesses. Able to reflect in a limited way on the project. **Band: 30-39** Unsatisfactory. Some attempt has been made at the background reading but clearly, only a partial understanding of the project topic. An incomplete attempt at the core tasks. Weak technical competence. Little ability to reflect adequately on the project. **Band: 20-29** Inadequate background reading but shows some limited understanding of how ideas can be linked to the task. A minimal attempt at the core tasks, showing poor understanding. A substantial amount of work is still needed to achieve the core tasks. Minimal reflection on the project. **Band: 10-19**

	A minimal attempt at background reading, inappropriate use of material, and almost no attempt at core tasks. Very poor understanding of the problem. Minimal or no reflection on the project. **Band: 0-9** No or almost no significant attempt
Remarks	There are no restrictions for using publicly available code and/or formulations (subject to correct referencing); however, grading is still subject to your own contributions. Nevertheless, this should not discourage you from utilizing the advanced techniques from the literature as long as you can demonstrate a sufficient understanding of the relevant scientific literature. If in doubt, talk to the module convenor. If you struggle with the implementation, you may still achieve a reasonable mark for the report explaining your solution method.
Academic misconduct	This is an individual assessment that should consist of your own unaided work. While it is good practice to use your own word when explaining something from a primary source, if you are unable to paraphrase the statement, then you should quote the statements and cite the source accordingly. If you are building on someone else's code (e.g. our code from the classes, open-source projects, etc.), please make it clear which aspects of the code are your work through the use of comments. The University has detailed advice about academic integrity, and submissions that demonstrate a lack of that integrity will be treated under appropriate disciplinary procedures. The academic misconduct policy document can be accessed via: https://www.nottingham.ac.uk/qualitymanual/assessment-awards-and-deg-classification/pol-academic-misconduct.aspx