

---

## Homework 1

Submission Deadline: **March 1, 2021 @ 23:59**

Please write the following at the top of your files for the programming assignment

- Names of both team members as it appears on LumiNUS
- Collaborators (write **None** if no collaborators)
- Source, if you obtained the solution through research, e.g. through the web.

### On Collaboration

- While you may collaborate, but you *must write up the solution yourself*.
- It is okay for the solution ideas to come from discussions. However, it is considered as plagiarism if the solution write-up is highly similar to your collaborator's write-up or to other sources.

### Submission

- Your solution for the programming assignment should be submitted to both LumiNUS (zip files) and **the code uploaded to aiVLE evaluation server**.
  - **Late submission:**
    - **Deadline for the written component is fixed**, no late submissions are allowed.
    - For the programming assignment, you will incur a late penalty of 20% of your score for late submissions.
    - No submission will be accepted after March 5, 2021 @ 23:59.
- 

### Programming Assignment (8 marks)

In this task, we will learn to generate the PDDL description files which will be used to solve 2 different planning problems. Before proceeding further, follow the instructions below to complete the setup required for this programming assignment.

### Installation Instructions<sup>1</sup>

- Setup docker on your machine, instructions for which can be found [here](#).
- Pull the docker that we have already setup for you with all the required dependencies. You can follow the instructions [here](#) to do so.

---

<sup>1</sup>The files to be installed are very large. Please be mindful if you intend to use metered data connections. Alternatively, you can do the installation if/when you are in SoC.

- After pulling the docker image, test your your installation by running

```
docker run -it --rm -v $PWD:/workspace cs4246/base python test_installation.py
```

for Linux/Mac or

```
docker run -it --rm -v \%(cd)\%:/workspace cs4246/base
```

python test\_installation.py for Windows (file given along with this assignment) on the docker container.

## Getting started

We will be using the `gym_grid_environment` (<https://github.com/cs4246/gym-grid-driving>) to simulate the solution obtained on feeding the PDDL files generated to a planner. These dependencies have been installed in the docker image “cs4246/base” which you have downloaded.

Read through the [introduction](#) and [example](#) from [Pellierd/pddl4j tutorial](#) to understand the PDDL description format. You can look at sample [problem](#) and [domain](#) files for further understanding. If you want to, you can also feed any problem and domain PDDL files to a planner by running the following command on the docker container (using the `docker run ...` command):

```
/fast_downward/fast-downward.py domain.pddl problem.pddl --search "lazy_greedy([ff()], preferred=[ff()])"
```

Note that in this PDDL format, specifying negative literals in preconditions is allowed.

For you to get used to the PDDL format, we have prepared a sample python script to generate PDDL files for the Air Cargo problem. You can run the file `pddl_cargo_example.py` (using the `docker run ...` command)] to see the PDDL files generated `cargodomain.pddl`, `cargoproblem.pddl` and also relevant portions of the code that generates it. Specifically, it will be helpful to look at functions :

1. `generateDomainPDDLFile()`,
2. `generateProblemPDDLFile()`
3. `generateInitString()`
4. `generateGoalString()`

To complete this assignment, you need to know some details about the environment : how a state is represented, the actions the agent can take, and few other small intricacies. We have prepared an IPython Notebook file on Google Colab [here](#) to walk you through all such details. It would be helpful for you to go through it before working on the tasks. While this may seem like a lot to understand for one homework assignment, we will be using the same environment for the other homeworks and for the project!

You are now ready to begin solving the two tasks which are listed below.

## Problem statement

### 1. Parking Task :

You are employed as a valet at a parking lot. You have to drive the car given to you and park it at a parking spot assigned to the car (identified as goal state in the environment). But, you also have to plan your way from your current position to the spot while wasting the least fuel possible (your tip depends on it!).

By virtue of being a Computing student, you decide to put your *planning* skills to use. You first retrieve an old python script written for a similar task by the TAs of CS4246, parts of which have been lost. The script used to generate the PDDL files and fed them to the fast-downward solver to generate a solution for this planning problem. It also runs the plan on the simulator to see if it does the job.

The script `python __init__.py` is missing 3 code snippets (marked with “FILL ME” in the file) .

- (a) Code which generates the action schemas of the three actions available namely, UP, DOWN and FORWARD. These action schemas should reflect the 3 actions available in the environment simulator. The agent’s speed range in the environment is restricted to  $[-1, 1]$  (negative speed moves towards the left, see the [environment](#) for more details).
- (b) Code which generates the initial state condition.
- (c) Code which generates the goal description.

You can test your code with the test configurations given in the script by running `python __init__.py` (present in the .zip file) on the docker (using the `docker run ...` command). **It might be helpful to look at the generated PDDL files for debugging.**

### 2. Crossing the road :

A customer forgets directions to the parking lot, and ends up on the other side of the road. The customer is not skilled enough to cross a busy multi-lane road with **moving cars**, where the speed of cars can differ across lanes, but cars in the same lane move with the same speed. To reach the entrance of the parking lot (identified as goal state in the environment), he calls you up and asks you to drive the car from his spot (the initial state) to the destination. Unlike the previous task, the agent’s speed range in the environment is restricted to  $[-3, 1]$ .

You decide to modify the script you wrote in Task 1 to handle this situation. You can test your code with the test configurations given in `__init__.py` (Replace the existing test cases with the one for the Crossing Task) .

**Hint:** Instead of modeling the parking lot/road as a 2 dimensional grid, it might be useful to include time as an additional dimension.

## Submission Instructions

Please follow the instructions listed [here](#). You have to make 2 separate submissions as mentioned below :

1. For Task (a) : You have been provided with a .zip file in the correct submission format. Complete the functions marked with “FILL ME” in `__init__.py`. Your submission zip file should have the exact same structure as the zip file you received.
2. For Task (b) : Modify the `__init__.py` as required and make a separate submission.

Submit both the zip files to the designated folder on LumiNUS as well.

Please name your files according to this format:

`<GroupName>-Task1` and `<GroupName>-Task2`.

E.g., `CS4246Group1-Task1.zip` and `CS4246Group1-Task2.zip`

**Note :**

1. Please remember to set the variable `SUBMISSION` in `__init__.py` to `False` when testing locally, and to `True` before making a submission.
2. Please do not print anything to the console, as it might interfere with the grading script.