# Homework 3 - Games

## Introduction

In this exercise, you take the role of the head of one of the two competing taxi agencies. Your goal is to bring as many of the passengers to their destinations as possible in the presence of a rival that tries to do the same!

## Environment

The environment is similar to the one in Exercise 1: The same grid world, with the same passengers and taxis. Key differences include:

- There are two separate fleets of taxis, one for each player.
- The actions are taken in turns.
- The map (i.e. the positions of passable and impassable tiles, as well as the size and initial passengers) is fixed.
- New passengers spawn on the map with some probability as the turns progress.
- The taxi fleets were upgraded to an environmentally friendly nuclear fusion design. Thus, they do not need fuel anymore, and no gas stations are present in the environment

Some features from Exercise 2 are also present:

- The agent is not required to output a complete plan but rather to output an action based on a state.
- The execution has a limited number of turns.
- The goal is to collect as many points as possible (more on that below).

## Actions

Actions remain the same as in Exercise 1 (“move”, “pick up”, “drop off”, “wait”), with identical conditions and effects, except for “refuel”, which is irrelevant. Note that the actions added in Exercise 2 (i.e “reset” and “terminate”) are also irrelevant to this exercise.

## Dynamics of the system

The game consists of two rounds. At the start of the round, each player is assigned a fleet of taxis they control, they are initialized, and then the turns take place. At each turn, one of the players decides upon their actions, returns them, and the resulting state is forwarded to the second player to make their decision. This happens continuously until the maximum amount of turns is reached.

In the second round, the players switch fleets. This is done to put the players on more equal ground, and not discriminate against any of them. After the second round is finished, the player who has the higher total score (sum of scores for the two rounds) wins.

## Points

Each passenger now has an additional trait - the number of points they provide to the player who drove them to their destination. This is the only source of points in this exercise. Your goal is to collect the maximum amount of points by dropping the passengers off at their desired locations

## The Task

Your task is to implement two agents:

- UCT-based agent - this agent should implement a UCT algorithm as taught in the lectures and tutorials. The agent must use random rollouts. **Note**: the UCT tree node does not have to contain a state (there are too many states resulting from agents arriving), it may just contain the actions the agent took to get there.
- A general agent - this agent may implement whatever algorithm you see fit to achieve the best results. It may be just a copy of a UCT agent if you so wish.

## Code

Your agents reside in hw3.py and for each, you have to implement two functions for each agent:

- A constructor `__init__(self, initial_state, player_number)`, which takes the initial state, your taxis, and an indication of whether you go first or second - should finish within 60 seconds.
- Function `act(self, state)` - takes in the current state of the game and returns an action. Should finish within 5 seconds.
- For the UCT agent only you must implement 4 additional functions:
  - Selection - `selection(self, UCT_tree):` - must select a node to expand
  - Expansion - `expansion(self, UCT_tree, parent_node)` - must update the tree with a new node
  - Simulation - `simulation(self)` - must perform a simulation from a new node
  - Backpropagation - `backpropagation(self, simulation_result)` - must update the UCT tree with the new information

The code consists of 5 files:

- ex3.py - The file with both of your agents. This is the only file you should modify.
- main.py - The file responsible for playing the game.

- utils.py - Contains some utility functions. You may use the contents of this file as you see fit.
- simulator.py - The file contains the simulator of the game. You may use it for your agents
- sample_agent.py - an agent that uses random actions

## Submission and grading

You are to submit **only** the file named ex3.py as a python file (no zip, rar, etc.). We will run the game against our own agents, and your ex3.py. The check is fully automated, so it is important to be careful with the syntax. The grades will be assigned as follows:

- Grading:
    - 85% for a correct implementation of the UCT agent.
    - 15% for a relative performance of your non-UCT agent compared to our bots of increasing difficulty:
        - 5% for defeating a simple UCT agent as you are required to submit,
        - 5% for defeating a UCT agent with a light heuristic rollout - using h1 from Homework 1.
        - 5% for defeating a UCT agent with heavy heuristic rollout.
        - The code of the agents will not be provided
- The submission is due on 26.01.2023 (the last day of the semester), at 23:59 - no summary extensions will be given as we are explicitly forbidden to extend the deadline past the semester except for miluim, parents, and hospitalizations.
- You still can submit the exercise up to 3 days late with a penalty of 5 points for each day.
- Submission in pairs/singles only. You may switch partners from HW1 and HW2 if you wish to do so.
- Write your ID numbers in the appropriate field (self.ids in ex3.py **agent class**) as strings. If you submit alone, leave only one string.
- The name of the submitted file should be “ex3.py”. Do not change it.

## Important notes

- We encourage you to double-check the syntax of the output as the check is automated.
- You may use any package that appears in the standard library and the Anaconda package list, however, the exercise is built in a way that most packages will be useless.
- You may import utils.py and use any function/class from there
- We encourage you not to optimize your code prematurely. Simpler solutions tend to work best.