Assigned: 12th of May 2025	**MAS HW 2 – Coding Assignment**
Due: 26th of May 2025	**LLM Agents in Auctions and Negotiations**

1. Homework Description

The following Homework is an individual coding assignment.
The goal of the homework is to explore the use of Large Language Models (LLMs) as the reasoning engines for agents that collaborate through the use of Auctions and Negotiation protocols.

Specifically, consider the following scenario:

A company (ACME) wants to build a new headquarters and needs to sign contracts for the following items: (i) structural design, (ii) building, (iii) electrical and plumbing work, (iv) interior design.
The contracting occurs by the following assumptions and rules:

- *The contracting company is called ACME and it has a given budget each construction item*
- *The contractors are companies A – F. Each company is specialized in a subset of the 4 construction tasks (e.g. structural design and interior design, building and electrical/plumbing work, etc). Each company has a lower limit for the price at which it will take on each type of construction task – its actual cost.*

- *The **primary goal of ACME** is to **complete the headquarters**. Its **secondary goal** is **to save as much money** as possible from the overall budget.*

- *The **primary goal of each contractor company** is to **participate in at least one contract**, since this builds up serious reputation. The **secondary goal** is to ensure it **turns in as much of a profit as possible**.*

To perform the contracting for each task, ACME employs the following strategy:

- *It first holds a variation of the Dutch (descending auction) (see Negotiation lecture) to establish a preliminary list of companies who are willing to do the task at a given price. The variation is that ACME will start from a low price and continuously raise it, until one (or more) companies place a bid. ACME is allowed to raise the price at most 3 times.*
- *With each interested company, ACME enters a negotiation using the monotonic concession protocol (see Negotiation lecture) to try and bring the price further down. ACME holds at most 3 rounds of concession. After the negotiations, it will select the company with the lowest price (or one randomly if each has the same offer).*

The modeling follows these indications:

- Model ACME and each of the contractor companies as an agent.
- **The environment** drives the interactions, keeping a record of each agent and knowing what services they can provide (i.e. structural design, building, electrical and plumbing work, interior design).
  - It maintains a state of two different interaction phases: auction phase and negotiation phase.
  - In the auction phase, the environment keeps track of current auction item, current auction round, companies that have secured a spot to the negotiation phase
  - In the negotiation phase the environment will facilitate a **Monotonic Concession Negotiation** for each of the agents that were selected by ACME during the auction phase:
    - **ACME is always the initiator of the negotiation**. ACME will try to start at lower price, and bring it back up to the value of auction phase.

- **Company agents are always the responders in the negotiation protocol**. Company agents decide if they accept lowering the received budget for the construction item.
- Company agents **do not know** the offers made in the negotiation by other agents, but they **do know** how many agents have entered the negotiation.

2. Support code

The support code provides engine to drive the sequence of interactions.
The **environment** module implements a step-wise interaction, starting with the reverse Dutch auction phase and followed by the Negotiation Phase. During each phase, each construction item is visited **in sequence** (i.e. the interactions for the "structural design" phase start and end **before** the interactions for the "building" phase start)

ACME and the contractor companies receive their configuration using yaml files.
The configuration of the game is given in the game.cfg yaml file, which specifies the number of auction rounds, the number of negotiation rounds as well as the list of agents taking part and their implementation reference.

The code stubs for the agents that you have to implement is in the **student_agent** module.

3. Task [10p]

Your task is to implement the reasoning that ACME and the contractor company agents make at each turn of the Auction and Monotonic Concession Negotiation stages.
To do so, you will use OpenAI's API to make calls to the **gpt-4.1-mini** or the **gpt-4o-mini** LLM model.

Guidelines:

- General:
  - Implement each agent reasoning step as a call to a *client.chat.completion* to an OpenAI() client
  - Define a **structured output** (see an example here) to ensure that the LLM returns a parseable output which you can easily parse (see an example here)
  - In the call to *client.chat.completion* set the *temperature* parameter to 0 and provide a *seed* value for a more consistent output on repeated trials
- Interaction specific:
  - Create separate prompting strategies for the two *interaction stages* (i.e. 1. Reverse Dutch Auction and 2. Monotonic Concession Negotiation)
  - For each stage:
    - Create a *system role (*see OpenAI examples*)* text prompt that specifies the *role* (e.g. ACME vs contractor company) the agent plays and the *general setup of the current interaction stage* (i.e. Auction or Monotonic Concession Negotiation). Be sure to include the details about the names of the construction stages, the number of participating companies, the rules of the protocols (e.g. the number of rounds), the budgets (for ACME) and costs (for contractor companies) for each construction phase, and the goals of each entity
    - Create a *user role* text prompt that will describe the current status of each specific interaction stage. Use a **structured format** to give information about aspects such as:
      - For the auction stage: the currently auctioned construction phase, the current round, the maximum number of rounds, the maximum budget / minimum cost, the previous budget offer (for ACME), the accepting companies, how many contracts are currently won (for a contractor company)

        - For the negotation stage: the currently negotiated construction phase, the current negotiation round, the maximum number of rounds, the last offer (for ACME), the last counter-offer (for each contractor company), the number of participating companies, the number of won contracts (for a contractor company), the number of negotiation items left (for a contractor company)
    - In the *user role* prompt, ask the LLM to "Think about it step by step, analyzing advantages and risks of its current situation"
    - In the *user role* prompt, **ask the LLM** to return the information you need to proceed with the protocol **using the provided schema.**

HINT:
Before you run the API calls, simulate a subset of the prompts you would use in the game **using the ChatGPT interface. Use separate chat windows for each prompt** to simulate independence of agent reasoning steps (by default, ChatGPT appends the output of the previous interaction to the *context* of the next prompt).
Does the reasoning of the LLM match with what you were expecting? Can you improve the prompt to modify the reasoning strategy?

TASK:
After implementing the API calls that build the agent reasoning:

- **Store the ensuing list of prompts that you submit to the LLM engine in a text file.**
- **Run the prompts in the order they would have been played in the protocol *in separate instances* of the OpenAI Playgorund interface and *retrieve the reasoning* that the LLM makes. Use the same LLM model (4.1-mini or 4o-mini) as you did in the programmatic exercise.**
- **Append the justification and the resulting JSON output in a text file, indexed by *interaction stage* (Auction or Negotiation), *interaction round number* and *company taking turn***

Submission content. You must submit two files:

- An archive containing your implementation based on the provided support code
- A PDF file in which you include:
    - An explanation of your strategy of engineering the prompts
    - The output of the **OpenAI Playground** generated justification and JSON outputs for the prompts you used in the game (see above task description)

4. Bonus [2.5p]

Implement the mechanics for an instance of the Monotonic Concession Negotiation Stage (i.e. the interactions for **one** construction item) using the LangGraph LLM agent interaction framework.

Specifically, model the Negotiation as an instance of a Graph with 3 types of nodes:

- The negotiation node (maintains state of the monotonic concession negotiation interaction)
- ACME agent node
- Company agent node(s)

An example of graph setup can be seen here.

Model two hypothetical scenarios, for **a single construction item** negotiation. Use budget and cost values from the given configuration files. The scenarios are:

1. ACME and a **single** company that has reached the monotonic concession negotiation stage
2. ACME and **two companies** that have reached the monotonic concession stage. One company already has an previous contract (for a different construction item), while the other one does not.

Define the roles of ACME and the company agents using the *create_react_agent()* prebuilt agent setup method.
Use the prompts developed for the main homework task and provide them to each agent using the make_system_prompt() call. Use a custom graph *state* class to model the current negotiation status when you pass control in your graph from one agent to the other.

Compare the outcome of the interaction in LangGraph with that of the controlled interaction from the main homework task, verifying if the agents you selected for the LangGraph interaction reach the same consensus as in the complete game.