Rivers and Stones

Assignment 2

September 9, 2025

1 Introduction

For Assignment 2, we will use the game of Stones and Rivers. The game is based on using stones to score in the opponent's Score Area while strategically positioning Rivers to enable sweeping movements across the board. Stones serve as the scoring side of each piece, while Rivers act as the mobility side, guiding Stones along their flow direction. For the purpose of the assignment, there shall be a total time for all moves played by the bot in the game.

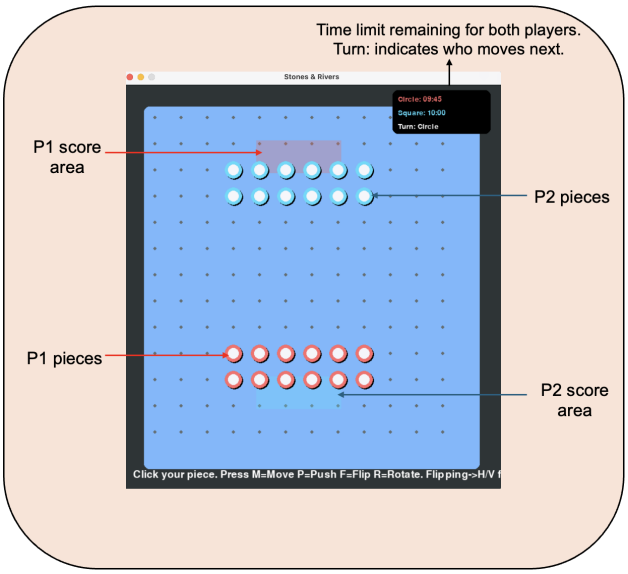


Figure 1: Example setup: A fully prepared board ready for the game.

1.1 Game Setup

- **The Board**
  The game is played on a 12 × 13 grid, which includes 24 starting positions (12 for each player) and two Score Areas. Each Score Area contains 4 spaces where Stones must be placed to achieve victory (see Fig. 1).

- **The Pieces**
  Each player begins with 12 identical pieces, but every piece has two sides: a Stone side, which is used for scoring, and a River side, which allows pieces to travel across the board through sweeping directional movement.

– **The Stone Side**: A piece with no mark in-between is called a stone.
– **The River Side:** A piece with a thin-width line in-between is called a river. The line can be horizontal or vertical, indicating the direction of the river (see Fig. 2).

1.2 Objective

A player is declared as winner, if

- You place 4 of your stone sides in your score area (present near the opponent's pieces at the beginning), with the stone face-up (not the river side), before your opponent achieves this goal.
- If your opponent runs out of time, and none of the players has successfully achieved the above goal, then you are the winner, and vice-versa.

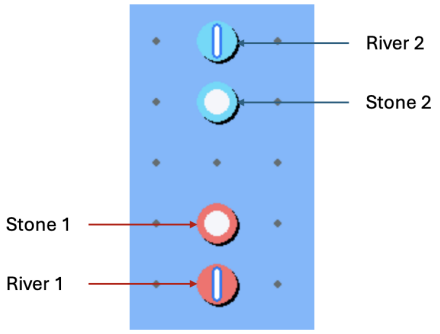


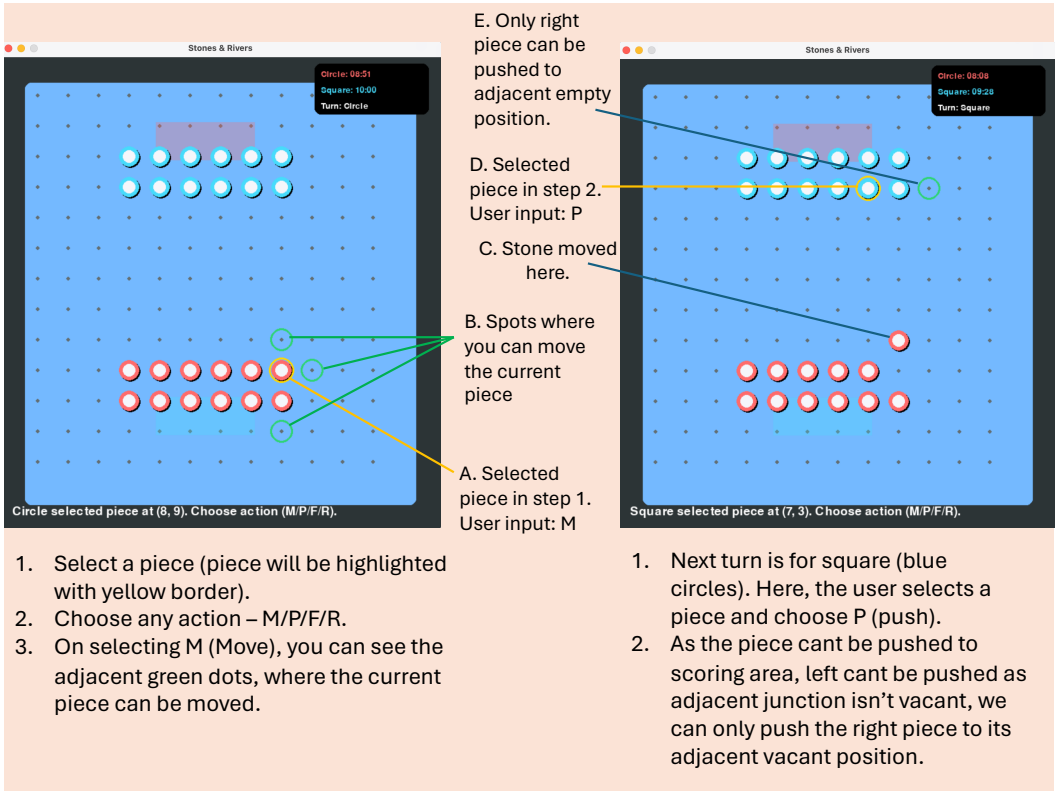
Figure 2: River and Stone pieces

1.3 Game Play Overview

On your turn, you may choose one action from the following four options:

- Move a Piece (Stone or River).
- Push a Piece (using either a Stone or a River).
- Flip a Piece and Rotate (switch Stone to River or reverse order, choose orientation if River).
- Rotate a River (change its flow direction).

1. **Movement Rules**
   - **General Movement**
     – Both Stones and Rivers can move.
     – Pieces move exactly one step up, down, left, or right, however, there are restrictions.
     – Movement restrictions:
       * Pieces must be placed on the intersections of the grid (all the dots visible are intersection of the grid, no half-movements in between these points are possible.
       * Cannot move into or pass through the opponent's Score Area.
       * Cannot leave the board.
       * Cannot stack multiple pieces on the same intersection.
     – See Fig. 3 to understand with an illustrative example.
   - **River Movement**
     – If you step onto a River, it may travel any number of spaces along the River's flow direction.
     – Travel stops when encountering another piece with stone side up (yours or your opponent's).
     – Both players' River pieces can be used for movement.
     – If you land on another River, continue moving in its new direction.
   - **Clarifications**
     – Stone and river both pieces can use River movement.



1. Select a piece (piece will be highlighted with yellow border).
2. Choose any action – M/P/F/R.
3. On selecting M (Move), you can see the adjacent green dots, where the current piece can be moved.

1. Next turn is for square (blue circles). Here, the user selects a piece and choose P (push).
2. As the piece cant be pushed to scoring area, left cant be pushed as adjacent junction isn't vacant, we can only push the right piece to its adjacent vacant position.

Figure 3: Illustrating move and push. As the rules describe, during move, we cannot jump to a non-vaccant cell, nor inside our scoring area.

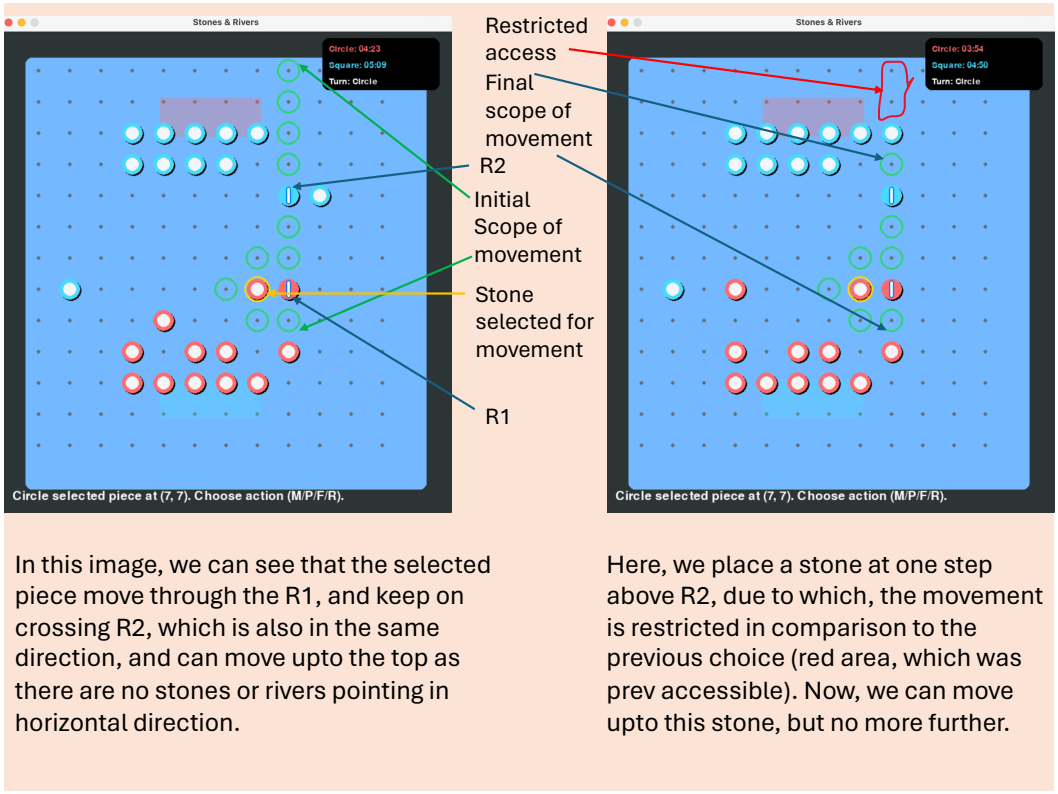


Figure 4: River movement with restriction example.

– You may continue the movement until blocked by another piece or you reach the end of the board; there is no hard-and-fast rule upto which you want to continue or where you want to stop.
– River movement cannot enter or cross the opponent's Score Area. (see Fig. 4)

2. **Pushing Rules**

- **Pushing with a Stone Piece**
When a Stone moves into a space already occupied by another piece, it may push that piece(stone side) forward by exactly one step in the same direction. This push is only valid if the next intersection is empty and lies within the boundaries of the board. A push is considered invalid if the displaced piece would be forced off the board, moved into the opponent's Score Area, or has no valid move in the chosen direction. In addition, Stones cannot push a series of pieces lined up together—only a single piece may be pushed at a time (see Fig. 5).

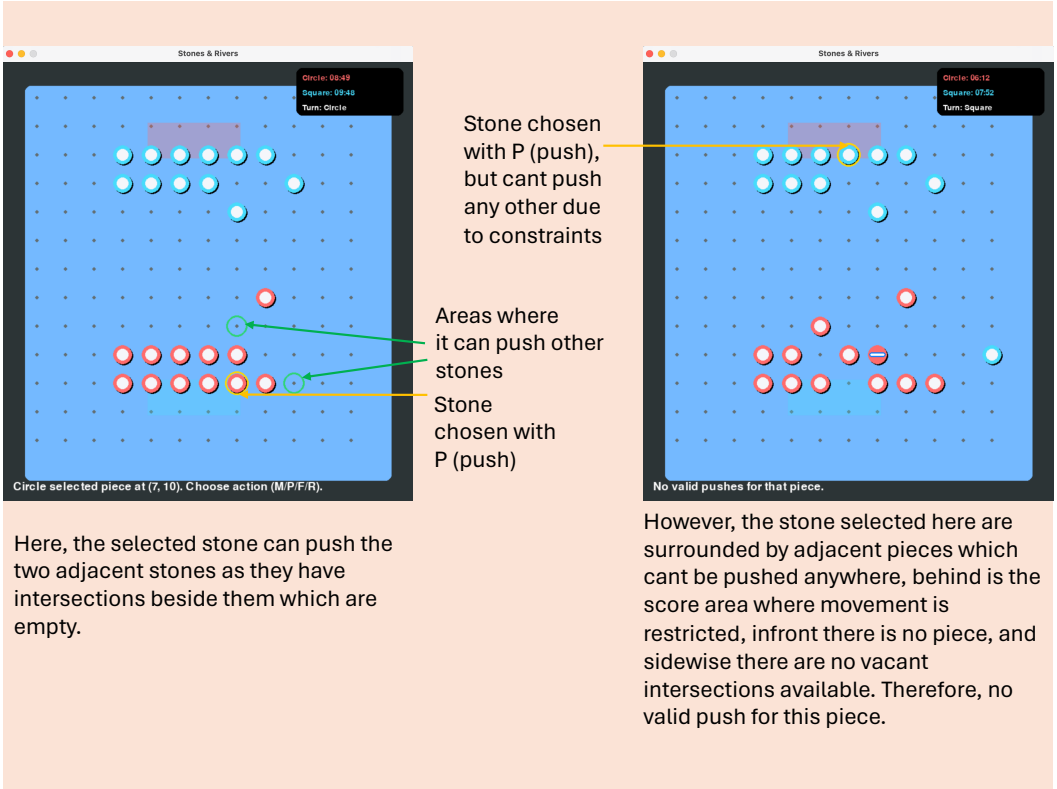


Figure 5: Illustrating pushing with a stone piece.

- **Pushing with River Piece**
Rivers allow a more flexible form of pushing. When a River moves into an occupied space, it can push the other piece any number of spaces along one of its flow directions, with the distance chosen by the active player. The pushed piece must follow the rules of River movement during this displacement and cannot push other pieces further along its path. After the push is resolved, the River that initiated the action is flipped over to its Stone side. Both your own pieces and your opponent's may be pushed in this way, and the displaced piece must be a Stone only. However,

the push is invalid if no valid landing space exists or if the resulting move recreates the exact same board position as the previous turn (see Fig. 6).

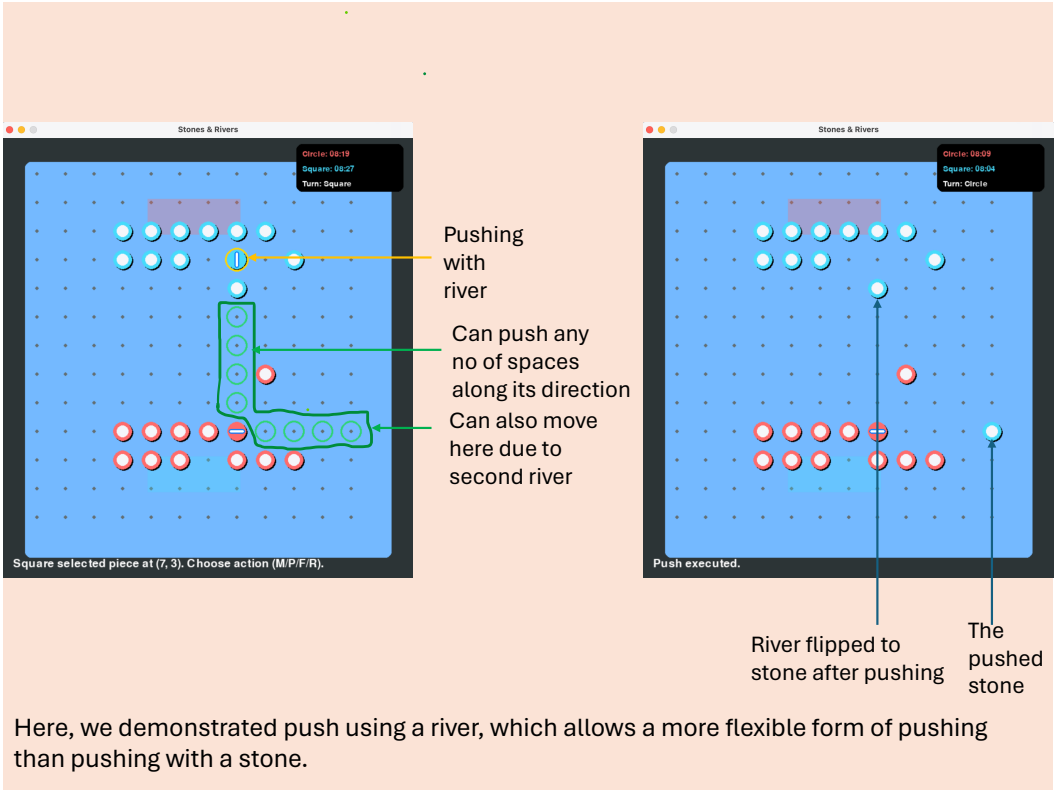


Figure 6: Pushing with a river piece, illustrating the difference wrt pushing with a stone piece.

3. **Rotating a River** Instead of moving, a player may rotate one of their River pieces by 90 degrees, switching it from vertical to horizontal or vice versa. Rotation cannot be skipped by keeping the same orientation, nor can it be rotated by 180 degrees. This action consumes the entire turn.

4. **Flipping and Rotating a Piece** Players may flip one of their pieces at any intersection. A Stone flipped into a River further requires choosing its orientation (vertical or horizontal), while a River flipped into a Stone becomes a scoring piece. This action also takes the whole turn (see Fig. 7).

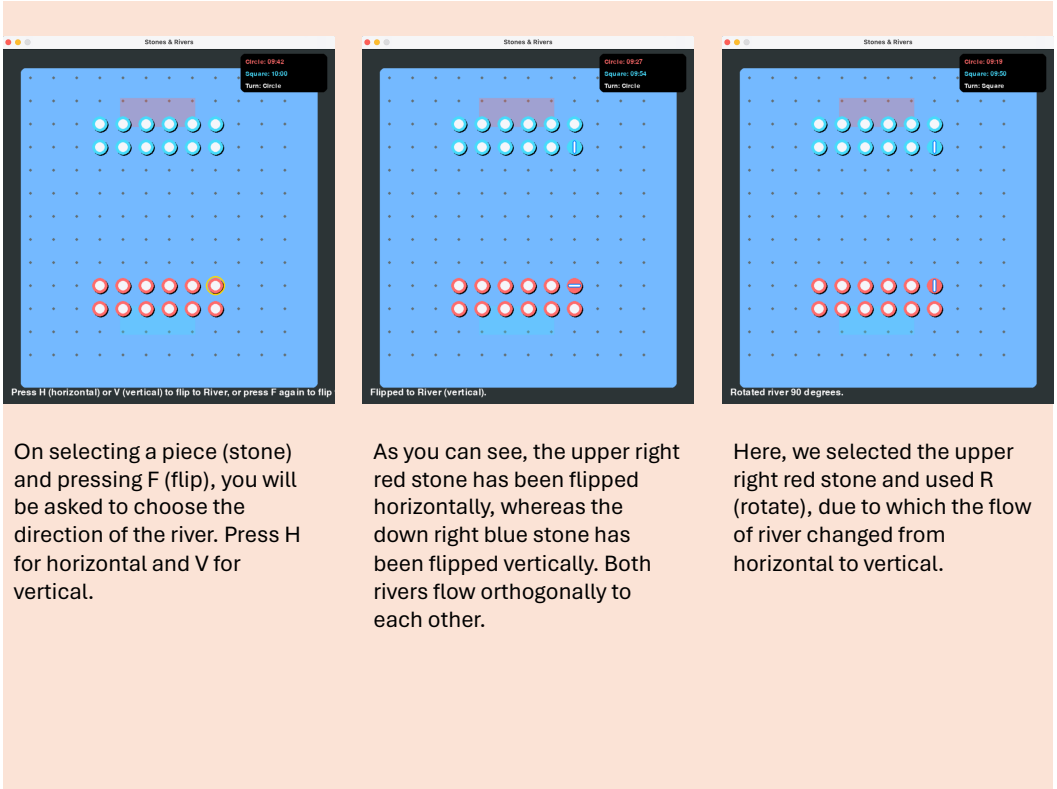


Figure 7: Flipping a piece and rotating a river.

1.4 Scoring Conditions

- Only Stone sides can score.
- Pieces in the Score Area remain in play. They may still be flipped, rotated, or pushed.

2 Scoring

Your final submission will be run against 2-3 TA Bots. Each game will be played twice, once with you as the circles and once as squares. You will be given 1 minute for the entire game for your moves during evaluation. Your final score in assignment is the based on the performance in these 4-6 individual games. The scoring pattern for each game is described below.

2.1 Game Ending conditions

The game is considered to be ended in either of the following scenarios:-

- A player wins immediately upon placing 4 Stones in their Score Area.
- The timer runs out for both players, the game will be treated as a draw.
- If both the players have played maximum number of allowed moves (500 here), the game will be treated as a draw.
- In case of timer running out for either player, other player will be treated as winner.

2.2 Total Score for a game ending in Victory

The winning player gets a victory score of 100 minus the score of the losing player. Let the number of pieces of the losing player with the stone side up in its scoring arena be n_{Lose} and the number of pieces (again with the stone side up) of the losing player that can reach their own scoring arena in one move be m_{Lose}. Then, the total score for either player is computed as follows. ($\mathbf{1}_{[X]}$ is the indicator function that is 1 when outcome X holds for the player and 0 otherwise.)

$$\text{Total Score} \;=\; \mathbf{1}_{[\text{Win}]}\,100 \;+\; \big(\mathbf{1}_{[\text{Lose}]} - \mathbf{1}_{[\text{Win}]}\big)\Big(n_{Lose} + \frac{m_{Lose}}{10}\Big) \tag{1}$$

2.3 Total Score for a game ending in a draw

$$\text{Total Score} \;=\; \text{Draw Score} \;+\; \frac{\text{Margin Score}}{4} \tag{2}$$

If the game draws, each player gets a **draw score of 30** (this is to discourage drawing of games) in combination with a margin score. Consider the following notation for calculating the margin score for the players:-

- n_{self}: Denotes the number of player's pieces, with the stone side up, that are in the player's scoring area.
- n_{opp}: Denotes the number of opponent's pieces, with the stone side up, that are in the opponent's scoring area.
- m_{self}: Denotes the number of player's pieces, with the stone side up, that can reach the player's scoring area in one move.
- m_{opp}: Denotes the number of opponent's pieces, with the stone side up, that can reach the opponent's scoring area in one move.

With the above notation, the margin score for either player is computed as follows.

$$\text{Margin Score} \;=\; 39 \;+\; \Big(\big(n_{self} + \frac{m_{self}}{10}\big) - \big(n_{opp} + \frac{m_{opp}}{10}\big)\Big) \tag{3}$$

3 Code Review

This assignment package provides the complete setup for running the game. The game server, implemented in Python, manages the flow of the match, while the client server includes starter Python code for your player. On the client side, you are expected to implement your AI bot, which will communicate with the game server to send moves.

The full codebase is available at : (Code) .Please refer to the README in the repository for detailed instructions on how to set up, run, and interact with the system. Note that the provided game server will also be used for evaluation purposes.

4 Interaction with Engine

The complete codebase for the game can be accessed through the repository link provided, which also contains a detailed README file with further instructions. The game has been developed using Python (version 3.x).

Installing Dependencies

A requirements file is included in the repository to facilitate installation. The dependencies can be installed by executing the following command:

```
pip install -r requirements.txt
```

Running the Engine

The engine is designed to operate in two modes: either through a Graphical User Interface (GUI) or via the Command Line Interface (CLI). The execution commands for the two modes are given below.

For running with GUI:

```
python gameEngine.py --mode hvai
```

For running with CLI:

```
python gameEngine.py --mode hvai --nogui
```

Write your code in student_agent.py and for running it you can either use

```
python gameEngine.py --mode aivai --circle random --square student
```

or you can use:

```
python gameEngine.py --mode aivai --circle random --square student --nogui
```

5 Submissions Instructions

- Submit your code for the Player in a zip file named in the format `<EntryNo1_EntryNo2>.zip`.
- Ensure that your entry numbers are correct (Kerberos IDs are not required).
- On unzipping, the following files must be present:
  - `student_agent.py`
  - `report.txt`
- Submissions not following this requirement will be penalized.
- Your code must compile and run on the provided VMs.

Report Requirements

First line of `report.txt` should start with this honor code. "Even though I/we have taken help from the following students and LLMs in terms of discussing ideas and coding practices, all my/our code is written by me/us." Second line should mention names of all students and LLMs you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you did not discuss the assignment with anyone, simply write `None`. You may include additional content after this line, but it will not be considered for grading.

Important

Your submission will be auto-graded. It is essential that your code strictly follows the input/output specifications. Any failure to comply will result in a significant penalty.

6 Phases Explanation

Phase 1 – Assignment 2 (A2)

1. This phase will carry **4 points** towards the overall course grade.
2. Your submitted **River and Stones bot** will be tested against **2-3 TA bots**. For each TA bot, your bot will play **two games** – once starting from each side of the board – making it a total of **4-6 games**.
3. A **seed** for each student bot will be assigned based on this final score, which will then be used for Phase 2. In case of a tie in final scores, the **tie-breaker** will be the cumulative scores of the TA bots.

Phase 2 – Assignment 5 (A5)

1. This phase will carry **12 points** towards the overall course grade.
2. Bots will be placed in a **tournament tree** based on their seeding from Phase 1 (similar to the Wimbledon Tournament format). The tournament will begin with a **round-robin stage** (4–6 bots per group) followed by a **knockout stage**.
3. The marks awarded to each bot will depend on **how far it advances** in the tournament. The exact scoring distribution will be announced later.
4. There will be **3 tournaments in total**, each worth **4 marks**. The winner of each tournament will be removed from the next ones and will receive the **full 12 marks**.
5. Further details and clarifications will be shared at the time of Assignment 5.

7 General Guidelines

- You can work individually or in pairs.
  - If working in a pair, include your team details in the write-up.
  - It is recommended to choose a partner you communicate well with, as this will help in the more open-ended final assignment (Phase 2).
  - You must continue with the same partner for the final phase, except if your partner drops the course. In such a case, you need explicit permission on Piazza to change partners.
- You must use python/c++ for designing your AI Agent. Note: Instructions to use c++ will be released in few days on piazza.
- Your submission must be entirely your own work.
- You cannot use built-in libraries/implementations for search or scheduling or optimization algorithms or gameplay. To request use of a library, use Piazza, and TAs will respond in 48 hours.
- Your implemented algorithm should be single-threaded. You are not allowed to make use of multiple threads, asyncio or any other form of parallel execution.

- You must not discuss this assignment with anyone outside the class. You cannot ask LLMs to write code for you. However, you are allowed to give the LLM your code in case you are stuck in debugging, so that you can learn about your mistakes fast, and do not waste time in debugging. Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team. Please read academic integrity guidelines on the course home page and follow them carefully.
- Please do not search the Web for solutions to the problem.
- Please do not use ChatGPT or other large language models for creating direct solutions (code) to the problem. Our TAs will ask language models for solutions to this problem and add its generated code in plagiarism software. If plagiarism detection software can match with TA code, you will be caught.
- Your code will be automatically evaluated. You get a minimum of 20% penalty if your output is not automatically parsable.
- We will run plagiarism detection software. Any team found guilty of either (1) sharing code with another team, (2) copying code from another team, (3) using code found on the Web, will be awarded a suitable strict penalty as per IIT and course rules.