# Snakes and Ladders

In this project, you are required to implement a game `Snakes and Ladders` using the Java programming language.

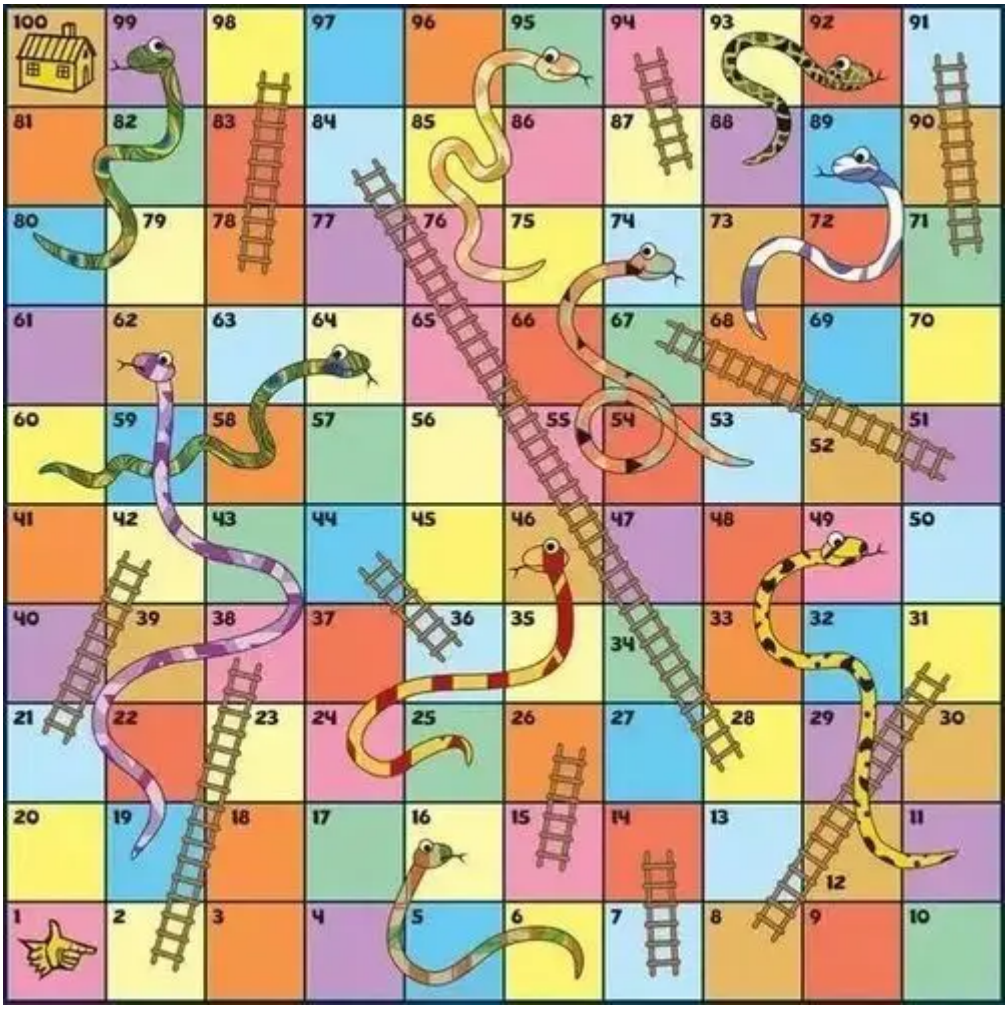
## I. Game Introduction

**Snakes and Ladders** is a classic board game for 2 or more players, originating from ancient India. The game consists of a numbered grid (typically 10x10), where players race to reach the final square by rolling a die.

**How to Play:**

- Players take turns rolling the die and move their token forward accordingly.
- Landing at the base of a **ladder** allows you to climb up to a higher square, speeding up progress.
- Landing on a **snake's head** forces you to slide down to its tail, setting you back.
- The first player to reach the last square (exact roll required) wins!

The game blends luck with simple strategy, making it fun for all ages.




## II. Game Interface & Basic Elements

**The game interface includes:**

- A **10×10 game board**, with each cell's serial number marked in the top-left corner.

- Each player controls a **game piece**, which must be displayed on the board.
- **Control buttons** for each player, used alternately to roll the dice, generate a dice number, and move the token.
- **Special cells** that trigger rewards (e.g., extra steps) or penalties (e.g., moving backward). To simplify the project, we use '+ steps' instead of ladders and '- steps' instead of snakes.
- A player wins **only if their game piece lands exactly on the 100th cell**.

# III. Detailed Game Rules

1. **Order of Play**:
   - Players roll the dice one by one, and keep the play order in all turns.
2. **Moving Your Token**:
   - Roll the die and move your piece **forward** the number of squares shown.
   - If you land on:
     - **Ladder (`+ steps`)**: Moving forward for `steps` steps.
     - **Snake/Chute (`- steps`)**: Moving backward for `steps` steps.
3. **Winning**:
   - You must **roll the exact number** to land on square 100. If the roll exceeds 100, your piece **bounces back** (e.g., if on 97 and you roll 5, move to 98\99\100\99\98, land on 98).

# IV. Project Requirements

In this project, you are required to implement multiple basic tasks that cover the fundamental functions of a Snakes and Ladders game, and you can also implement some advanced tasks to make your program more powerful. As we evaluate your program and find a task is implemented and the functionality is generally available, you will get the corresponding score.

## Task 1: Game Initialization (10 points)

1. After the user logs in or chooses the guest mode, the game will display the board.
2. The game should allow players to restart a new game at any time during gameplay. (Not exiting the program and run it again.)
3. When restarting a new game, the game data needs to be consistent with the new game.
4. **The game interface must include:**
   1. A game board
   2. Dice roll results display
   3. Individual roll-dice buttons for each player

   **The board must display these essential elements:**
   - Current positions of all players
   - Grid numbering (cell identifiers)
   - Special grid spaces (e.g., +N/-N steps)"

## Task 2: Multi-user Login (20 points)

1. Implement a login selection interface for both guests and registered users.
2. Guests can play without registration but do not have the functionality to save game progress.
3. The user login interface includes a registration page and allows login after entering account credentials.
4. After the program exits and re-runs again, previously registered users can still log in.

## Task 3: Save and Load Games (20 points)

1. Each user (except guests) has the option to load their previous saved game; the save is a single save file, and saving again will overwrite the previous save (Overwriting the original save is the basic requirement. Additional points would not be given if multiple save slots are implemented per user.)
2. From the game start interface, players can choose to load their last save which should contain information about the the game board's status and the number of moves made so far.
3. Each user's save data is unique.
4. Manual saving is a basic requirement; implementing automatic saving at timed intervals or upon exit can earn points in the advanced section.
5. Save File Error Check: If a save file's format or contents are corrupted when loading, the damaged save will not be loaded, and the game will still run rather than crash. (If your game is capable of detecting save files that have been modified by others while still maintaining the legitimacy of the save data，it will earn the advanced points.)

## Task 4: Gameplay (30 points)

1. **Order of Play**: The basic requirements specify a two-player game where players take strictly alternating turns to roll the dice. The program must enforce turn order by preventing the non-active player from rolling - when it's Player A's turn, Player B's roll button should be disabled, and vice versa. The system needs to maintain correct turn sequencing throughout the game session, ensuring Player A and Player B can only roll the dice when it's their respective turn..
2. **Each turn:** Each player clicks his/her own roll dice button and gets a random number from 1 to 6, which is the steps the player needs to move. Then the piece of the player should move steps forward.

   If the piece lands on special grids:

   - **Ladder (`+ steps`)**: Moving forward for `steps` steps.
   - **Snake/Chute (`- steps`)**: Moving backward for `steps` steps.

   Player must **roll the exact number** to land on square 100 to win the game. If the roll exceeds 100, player's piece **bounces back** (e.g., if on 97 and you roll 5, move to 98\99\100\99\98, land on 98).
3. **Game Victory**: Check if a player has landed exactly on square 100. If so, display the victory screen for that player. After the first player wins, the game can either: 1) Terminate immediately, or 2) Continue for remaining players (skipping the winner's turn) - both implementations can get the points of this requirement.

## Task 5: Graphical User Interface (GUI) (10 points)

1. Implement a graphical interface for the game using JavaFX, Swing, or any other **Java** graphical framework.
2. You will earn points for this section by completing the code based on the demo provided in the course.
3. Independently creating a GUI will count as Advanced points.
4. If your program needs to input into command line，you can not get full points of this task.

## Task 6: Advanced Features (30 points)

Any additional features beyond the basic requirements described above will earn points in this advanced category, including but not limited to:

1. Enhanced graphics and aesthetics
2. More game modes design
3. Adding some animated effects
4. Adding props in the game
5. Adding maps of your own design
6. ……

## V. Teamwork & Grading

You are required to work in a group of 2–3 members, where all members should be in the same lab section. You will get a *baseline score* by completing tasks (basic: up to 90 pts + advanced: up to 30 pts). We will then adjust the score based on the size of your team:

2 members: adjusted = baseline * 1.0

3 members: adjusted = baseline * 0.9

Members can adjust their *contribution ratio* within the group, which should be within the range of 0.8–1.2. The sum of all members' contribution ratio should be equal to the number of members. Contribution ratio adjustments need to be informed and agreed to by all members, teachers and TAs are NOT responsible for coordinating them.

Besides, this project will have two deadlines:

**Early Deadline** (Week 15): submit code to Blackboard before May 27, 23:59 & present in the lab session of Week 15. To incentive early submission, a 5% bonus will be awarded.

**Regular Deadline** (Week 16): submit code to Blackboard before June 3, 23:59 & present in the lab session of Week 16. All remaining groups must complete their work by this deadline.

**Personal Final Score = Min(110, Baseline * Team Size Adjustment * Contribution Ratio * Deadline Adjustment (1 or 1.05) )**

The personal final score should not exceed 110.

**Showcasing Your Work**

You need to demonstrate your work in a classroom-defense in the lab session of Week 15/16. All team members should be present.

The presentation should include a brief introduction of your project design, for example, how you persistent data, how you manage the state of CLI, etc. You may prepare a slide (PPT) to help you present your work, but it is not required.

Then you should run your program and demonstrate its functionalities. Please prepare your own flow in advance so that you can demonstrate the finished tasks smoothly and adequately. Teachers and TAs may ask questions during the whole process of your presentation, to verify the work's authenticity and to score more accurately.