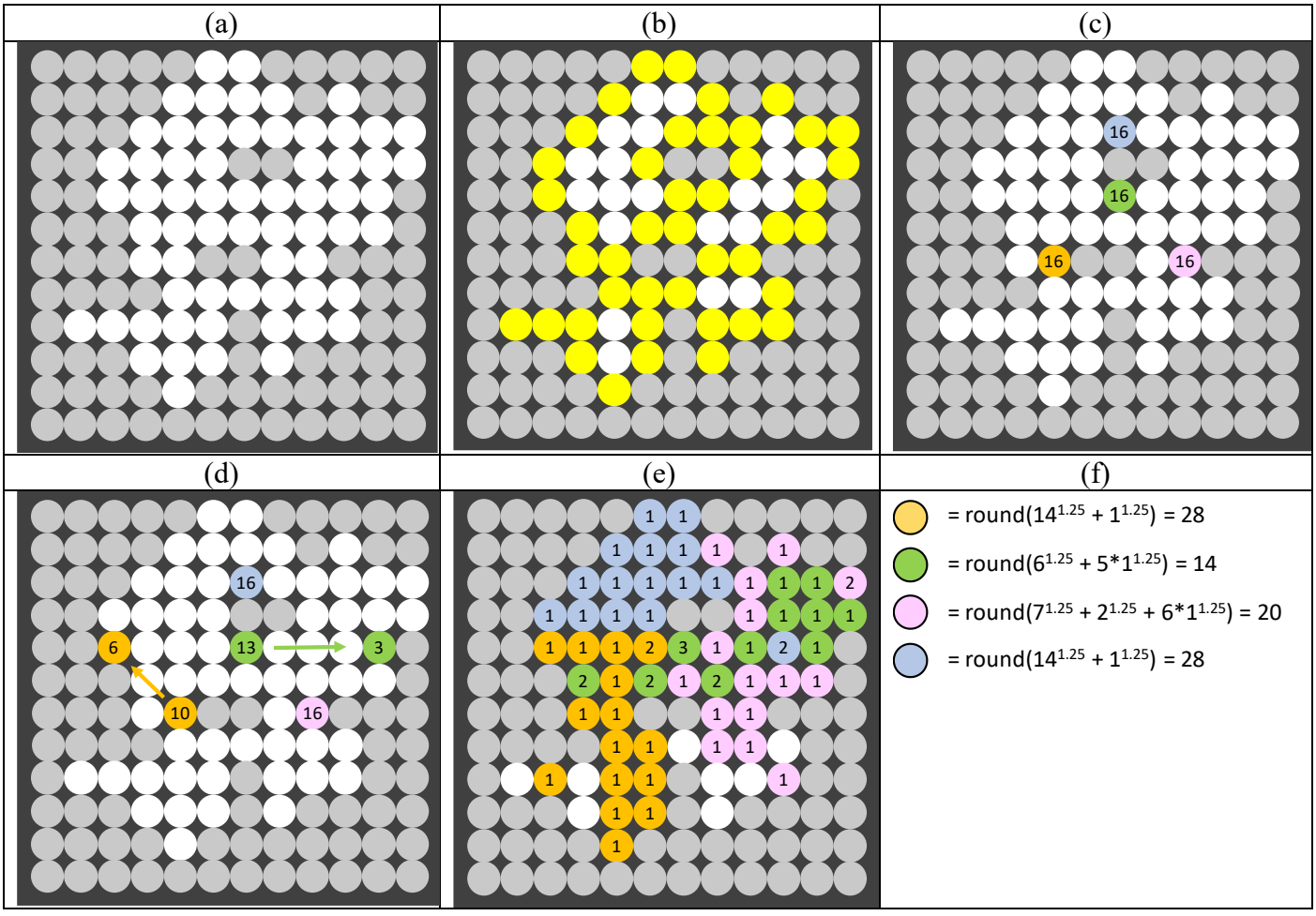
Undergraduate AI Capstone NYCU Spr2024 Project #2 Due 4/14/2024

The objective of this assignment is for you to design game-playing agents. This is a group project on a board game called Battle Sheep. The "game board" is just a set of circles (cells) arranged on a grid.

Description and rules of the game (basic form):

- Figure (a) is an example 12x12 board. The white circles (64 in total) are the actual playing field, which is randomly initialized for each game. The playing field always forms a connected region.
- Figure (b) is the example playing field with the border cells (cells in the playing field that are at the board boundary or adjacent to non-playing-field (gray) cells) colored in yellow.
- Each player is allocated 16 sheep. The first move of a player is to place all the 16 sheep in a single border cell. An example board after the first round is in Figure (c).
- For game state display, the color of a cell indicates its occupying player. The number in a cell indicates the number of the player's sheep in that cell.
- In each move, a player
  - Selects a cell with more than one sheep,
  - Splits those sheep into two non-empty groups,
  - Moves one group along a line of unoccupied cells as far as it can go (no stopping until it cannot move any further, and no hopping over occupied cells or non-playing-field cells), and leaves the other group at the original place.
- Example moves of two of the players are shown in Figure (d).
- A player has to "pass" (skip the move) if no valid move exists.
- Score of a player = round[SUM((size of each connected region of a player's cells)$^{1.25}$)].
- Figure (e) is an example terminal state of the game, with the resulting scores given in Figure (f).



Algorithm and implementation:

- You have a lot of flexibility in designing your game agent. It can be as simple as a set of rules. You can try the classical method of minimax search, possibly with alpha-beta pruning. You can also try to implement MCTS, or to train your agent using reinforcement learning. Since there are 4 players in the game, your game tree will grow such that each round consists of four layers. The decisions / evaluations made at each layer is based on that particular player's perspective.
- You are allowed to use modules/libraries developed by others for game playing. Even so, you learn the most if you try to understand how those modules are constructed as well as do experiments with what you can change in the modules. You also should specify in your report and programs the outside codes that you used.
- When the TAs run the tournament, the game server and all the player programs will run on the same computer. There will be no outside connectivity. There will be a time limit (3 seconds) for each move.
- You can only implement the programs in C++ or Python 3.

Game agents to submit, one for each game setting:

- Agent 1: The basic form (4 players, 16 sheep per player, 64-cell playing field)
- Agent 2: Overcrowded bigger board (4 players, 32 sheep per player, 100-cell playing field within a 15x15 square)
- Agent 3: Imperfect information: A player does not know the numbers of sheep in the cells occupied by the other players.
- Agent 4: Cooperative play: The four players form two pairs: Players 1 & 3, and Players 2 & 4. Each player is scored individually first, and then averaged with the score of its partner.

To be provided by the TAs:

- Environment and hardware spec.
- Game-play server programs (one for each game setting):
  - Game state representation
  - Game board initialization
  - Communication with player clients (via TCP)
  - Game management / time control
  - Scoring
  - Game recording
  - Graphical interface for playback
- Player client code template (Python3 and C++) and instructions
- NPCs (as opponents) for each game setting

Teams:

- The students should form teams of 1-3 members. A link will be provided for you to indicate your team.
- A team ID will be assigned to each team later. You need to include this ID in both your filename and within the code.

The tournaments:

- Four tournaments, one for each game setting, will be run by the TAs after the due date.
- For each game, the players with the 1^{st}, 2^{nd}, 3^{rd}, and 4^{th} highest scores receive 5, 3, 2, and 1 tournament points, respectively. When there are ties, the tournament points are evenly distributed among those players with tied scores.
- Each team's program will be played for the same number of games, with the same number of games in each of the 4 starting positions. The player combination of the games will be arranged randomly.
- Ranking is based on the total tournament points of the teams. Total game scores are used as tie-breakers for teams having the same total tournament points.
- There will be one ranking for each game setting.
- The rankings will factor into 50% of your grade for this project. (The other 50% is based on your report.)

Submission:

- The submission is through New E3. Your programs and reports will be submitted separately. No late submission is accepted for the programs. The reports are due on a slightly later date (**4/17**) and late submissions of up to three days are accepted.
- Follow instructions by the TAs for the submissions of your programs. The four agents will be submitted on separate entries on E3. The TAs will announce later whether you need to submit executables.
- The report (maximum 10 pages single-spaced) should describe how your game agents work, your experiments and experiences, similarities and differences in the different game settings, and contributions of individual team members.
- Remember to list the team name, ID, and members in both the source code and the report.