**CMPS 270 BATTLESHIP BOT**

**Player Struct Update**

* Add a bool to the Player struct to indicate whether the player is a bot or a normal player.
* Add a probability distribution grid

**Bot Ship Placement**

1. **Random Placement (Easy-bot mode)**
   * Bot places ships randomly on the grid.
2. **Probability-Based Placement (Hard-bot mode)**
   * The bot selects locations with the **lowest probability** of being hit.
   * A random number of ships will follow this logic, with at least **one ship** placed according to probability.
   * The remaining ships are placed randomly.

**Bot Firing Logic**

* The bot uses the already implemented attack methods and applies math to decide the best square to target.

**Bot Intelligence Levels**

1. **Smart Intelligence**
   * Maintains and updates a **probability grid**:
     + The grid is initialized once and updated each round after the bot fires.
     + Tracks the number of possible orientations for each non-sunken ship.
   * Probability Distribution:
     + For each non-sunken ship, probabilities are assigned to squares where it can fit. (the number of orientations the ship can be placed in)
     + Probability grids of all non-sunken ships are summed to update the grid.
   * Updating the Grid:
     + No need to fill the grid from scratch every time, just update the region where a change was made during the round.
     + For squares needing updates, recalculate probabilities based on the remaining ships.
   * Bot Weapon Usage:
     + **Artillery**: Targets the 2x2 region with the highest probability sum.
     + **Torpedo**: Targets the row/column with the highest probability sum.
     + Always uses **Artillery** or **Torpedo** when possible, choosing the most effective square.
   * Bot **does not use Smokescreen** for now.
   * Bot uses radar in 3 random turns on a high probability target (tba later)
2. **Average Intelligence:**
   * Chooses semi randomly using the checker method, and when it hits it continues hitting next to it till it sinks the ship
   * It chooses randomly what ship size to target first (by what amount of squares to jump every attack)
   * Sequentially sinks ships, it might be searching for the destroyer but hit the submarine and continue till it sinks the latter.
   * Picks regions in the graph where it thinks there’s the highest chance of a hit (could be probabilistic)
   * Adds every hit to a queue (adds it only if the ship wasn’t sunk yet; uses a specific method to check if the hit sank a ship)
   * When the queue is not empty, it pulls from the queue and targets squares next to that one according to how it thinks the ship is oriented.
3. **Dumb Intelligence:**
   * Picks randomly
   * Knows when a square can hold no ships and doesn’t pick it (not too dumb after all)

**Driver Code Update**

1. Check if the player is a bot:
   * If player.isBot == true:
     + Call PlaceBotShips() and pass the current player.
   * Otherwise, use the standard placement logic.