CECS 277 – Lab 10

Mind Reader

Create a program that predicts one of two choices that the user will enter. Prompt the user to enter one of two possible inputs: X and O, or a sentinel (ex. 'Q'), to allow the player to quit.

The program makes predictions based on the history of user's previous inputs by storing the last four inputs into a pattern string, and then using a HashMap to store the different patterns that have come up (the keys), with a count of how many times that pattern has appeared before (the value).

Every round, the program follows the same basic algorithm:

1. The computer makes a prediction of what value the user will choose by calling makePrediction (do not display this until the user enters their choice).
2. Prompt the user to enter their choice by calling getInput.
3. Compare the user's choice and the computer's prediction. If they are the same, then the computer gets a point.
4. Increment the number of rounds played and then display the computer's prediction and win percentage.
5. Store the user's choice into the pattern. Add the new input to the end of the string and remove the oldest one from the beginning (ex. if pattern = "OOXO", and the next input is 'X', then take the last 3 values from the pattern "OXO, and concatenate the new input onto the end to get the new pattern: "OXOX").
6. Store the pattern into the HashMap by calling storePattern.

Make the following functions:

1. makePrediction – pass in the HashMap and the pattern string. Use the last three values in the pattern along with the possible outcomes to check the HashMap to examine how many more times the user has chosen X over O after that pattern. Return the character matching the larger result. If there isn't enough information to make a determination, then return a random result.

   Ex: Make a prediction based on the following: pattern = XOXO, and HashMap = […, "OXOX" = 10, "OXOO" = 4, …], it compares the two values 10 and 4, and determines it's more likely the user will enter 'X'.
2. getInput – prompt the user to enter 'X', 'O', or 'Q' (lower case are also acceptable, just make sure to store all the same case into your patterns). Get the user's input and return it if valid, otherwise reprompt them for a valid entry.
3. storePattern – pass in the HashMap and the pattern string. If the pattern already exists in the map, then increment its value, otherwise, initialize it with 1.

For the first few rounds the computer will obviously not have enough information to make a prediction (makePrediction will randomly return one). But after that, it should catch on to any repeating pattern pretty rapidly. Try to enter values "randomly", the computer will usually be able to pick up on patterns that most people think are random.

Example Output:

```
Mind Reader!
Enter X or O, or Q to quit.
?: X
Comp: X
% Wins: 100.0
?: X
Comp: X
% Wins: 100.0
?: X
Comp: X
% Wins: 100.0
?: X
Comp: O
% Wins: 75.0
?: X
Comp: X
% Wins: 80.0
?: X
Comp: X
% Wins: 83.3
?: X
Comp: X
% Wins: 85.7
?: X
Comp: X
% Wins: 87.5
?: X
Comp: X
% Wins: 88.9
?: X
Comp: X
% Wins: 90.0
?: X
Comp: X
% Wins: 90.9
?: O
Comp: X
% Wins: 83.3
?: O
Comp: O
% Wins: 84.6
?: O
Comp: X
% Wins: 78.6
?: O
Comp: X
% Wins: 73.3
?: O
Comp: O
% Wins: 75.0
?: O
Comp: O
% Wins: 76.5
?: O
Comp: O
% Wins: 77.8
?: O
Comp: O
% Wins: 78.9
?: O
Comp: O
% Wins: 80.0
?: O
Comp: O
% Wins: 81.0
?: O
Comp: O
% Wins: 81.8
?: X
Comp: O
% Wins: 78.3
?: O
Comp: O
% Wins: 79.2
?: X
Comp: X
% Wins: 80.0
?: O
Comp: O
% Wins: 80.8
?: X
Comp: X
% Wins: 81.5
?: O
Comp: O
% Wins: 82.1
?: X
Comp: X
% Wins: 82.8
?: O
Comp: O
% Wins: 83.3
?: X
Comp: X
% Wins: 83.9
?: O
Comp: O
% Wins: 84.4
?: Q
Game Over
```