Problem 1

A symmetric game is one where any permutation to the order of the players (e.g. swapping players 1 and 2) does not change the game. For such permutations to have no effect, it must be the case that all players have identical action sets, and that the payoff to any outcome depends only on *how many* players select each action, not which specific players chose an action.

A corollary of Nash's theorem that appears in Nash's original 1951 paper is that every symmetric game has a symmetric Nash equilibrium. In a symmetric game, a mixed-strategy profile is symmetric if all players are following the same mixed strategy. To prove this corollary, we will take a different approach, based on the symmetric replicator dynamics function.

When running replicator dynamics on a symmetric game to search for a symmetric equilibrium, we don't need to keep track of separate strategies for each player. Instead, we can use a single mixed strategy, that will be played by all players, to represent the profile. We can then treat the update done on each round of replicator dynamics as a function that maps a mixed strategy to a mixed strategy. We will use this function to prove the existence of symmetric equilibria in symmetric games.

(a) Define as a mathematical function the update performed by one iteration of replicator dynamics.

Your Problem 1a solution goes here.

(b) Prove that a mixed strategy is a symmetric Nash equilibrium if and only if it is a fixed point of this function.

Your Problem 1b solution goes here.

(c) Use Brouwer's fixed-point theorem to show that this function must have at least one fixed point.

<div style="border:1px solid black; min-height:900px;">

Your Problem 1c solution goes here.

</div>

Problem 2

In the Jupyter notebook `nash_learning.ipynb` implement the following functions:

- `fictitious_play`

- `replicator_dynamics`

- `random_profile`

- `RD_random_restarts`

Note that the version of replicator dynamics your are implementing is the asymmetric version, and is therefore not identical to the function you used in problem 1. A function for plotting the traces of replicator dynamics and fictitious play will be available soon.