CHE 599, HW1: Monte Carlo simulation of the Monte Hall problem

due Monday, October 7th. upload your `.ipynb` to Canvas.

learning objectives

- familiarize with data structures, functions, arrays, control flow, programming paradigms in Julia
- implement Monte Carlo simulations of a stochastic process; we will use Monte Carlo to resample data throughout this course

the Monty Hall problem

Watch this YouTube video and/or read this Wikipedia page to learn about the Monte Hall problem. The Monte Hall problem has sparked debate and confounded statisticians and mathematicians for years, including famous mathematician Paul Erdos (see "Which Door Has the Cadillac?").

> Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?

the deliverables for this homework

In a single Jupyter Notebook, please write a Julia code to simulate the Monte Hall problem under two contestant strategies: (i) persist with your first door choice, (ii) switch choices after the game show host opens a door and reveals a goat. While it is possible to code this up in many different ways, I provided some structure below to encourage you to (i) explore the capabilities of Julia and (ii) modularize your code into functions that do independent tasks (good practice for organization, clarity/readability, verification, and future extendability).

1. The `shuffle!` and `sample` functions will probably be useful to you.

```
using Random # for the `shuffle!` function
using StatsBase # for the `sample` function
```

2. Write a `mutable struct Door` to represent a door with two attributes: (i) whether or not it has been opened by the game show host to reveal a goat and (ii) what is behind the door (a goat or a car). We make it mutable so that the game show host can open it.

3. Write a `function set_up_doors()` that constructs and returns an `Array` of three `Doors` (`Array{Door}`) such that:
   - two doors have a goat behind them
   - one door has a car behind it
   - which door has the car is chosen at uniform random
   - all doors are unopened
4. Write a `function pick_a_door_at_random()` that returns an integer in $\{1, 2, 3\}$ at random, representing the door that the contestant chose at random.
5. Write a `function open_a_door!(doors::Array{Door}, first_door_id_pick::Int)` that opens one of the `Doors` passed to it such that:
   - the door that the contestant picked, `first_door_id_pick`, is not opened
   - the door with the car is not opened
   - if two doors are viable to open, pick one at uniform random
6. Write a `function switch_door_pick(doors::Array{Door}, first_door_id_pick::Int64)` that returns the index in $\{1, 2, 3\}$ of the door that the contestant must pick if he/she switches from his/her initial door choice to the other unopened door. It is assumed in this function that one the `Doors` passed has been opened by the game host and has a goat behind it.
7. Write a `function contestant_wins_car(doors::Array{Door}, door_id_pick::Int)` that returns `true` if the contestant picked the door with the car behind it and `false` otherwise.
8. Write a `function simulate_Monte_Hall(switch::Bool; verbose::Bool=true)` that uses all of the functions above to simulate one Monte Hall game. And returns `true` if the contestant won and `false` otherwise. The `switch` argument specifies whether the contestant uses the strategy to switch doors. The `verbose` argument specifies whether the function (for debugging and illustration) prints off details of the outcome including (i) what is behind each door, (ii) what door the contestant picks, (iii) what door the game show host opened to reveal a goat, (iv) (if applicable) the choice of the door after the contestant switches, and (v) finally, if the contestant won or lost.
9. Run 1000 simulations of the Monte Hall problem under the two strategies. Print off how many times the contestant won under both of the strategies.

```
nb_sims = 1000
for s = 1:nb_sims
        simulate_Monte_Hall(false, verbose=false)
        simulate_Monte_Hall(true, verbose=false)
```

```
        # keep track of how many wins under each strategy...
end
```
