(Exercise 9) For this week's exercise I will provide some code below to help you reach Part II. Recall the Gambler's Ruin problem on the integers $\{0, 1, 2, \ldots, 10\}$ with bets at each stage of an amount $a \in [1, 2, \ldots, x]$, and a fixed (independent) probability p of winning each bet. A very useful way to formulate Gambler's Ruin has already been seen in the notes:

- to have no rewards at all, because instead we ...
- place a boundary condition on the *value function*, essentially enforcing that $V_n(x) = 1$ for any $x \geq 10$. (This is like receiving a reward of $+1$ after reaching an $x \geq 10$.)
- Note the reward is obtained after moving, not during the action, so step-rewards are 0 and also independent of the action taken.

What we therefore do is as follows: let $V_T(x)$ represent the maximized reward over T steps following an optimal betting strategy starting with wealth x. Then if during our T steps we reach a state ≥ 10 we will have collected reward of $+1$, if we haven't reached such a state (including all cases where we have gone bankrupt already) we have a reward of 0. As discussed in lectures this approach will maximize the probability of reaching such a state ≥ 10.

In part (h) of Exercise 9 you should use the following 'shell' for a function:

```
findValueFunction <- function(T,p){
  steps <- 0
  #Insert your code here creating V
  #Insert your code here creating W, putting ...
  #                          in boundary values
  while(steps < T){
    #Insert code here for putting in ...
    #                boundary values of W
    #Insert loop code here for putting ...
    #      values into W using Optimality ...
    #      Equation, V, and also storing the ...
    #      best choice a* for each state x

    V <- W
    steps <- steps + 1
  }
return(list(value=W,actions=<your.best.action.vector>)
}
```

findValueFunction(100,0.4), for example, will now work out $V_{100}(x)$ when $p = 0.4$. Can you see why? It even tells you which action to optimally take initially.

(The idea here is that the code increments t from 1, with V always containing V_{t-1} and being used (at each x) to work out V_t which is then stored in W. Then t is incremented, W stored into V and the process repeated. The value of t although not explicit, is always steps+1.)

Exercise 9 (Al-Khwarizmi)**. Part I:** Solving Gambler's Ruin.

Our value function for $V_T(x)$ satisfies this optimality equation:

$$V_T(x) = \max_{1 \le a \le x} \left[pV_{T-1}(x+a) + (1-p)V_{T-1}(x-a) \right],$$

which holds for all $T \geq 1$, and $x \in \{1, 2, 3, \ldots, 9\}$. We also have boundary conditions:

- For all $T \geq 0$: $V_T(0) = 0$ and $V_T(x) = 1$ for any $x \geq 10$;
- $V_0(x) = 0$ for $x \leq 9$;

You will use this Optimality Equation (for each $x \in \{1, 2, \ldots, 9\}$) to calculate V_n from V_{n-1} for each $n \geq 1$ until you have found $V_{1000}(x)$. (I suggest you fix $p = 0.4$ until Part II). Parts (a)-(b)-(c) are intended to be very quick, the exercise really starts at (d).

(a) Why is $\mathcal{X} = \{0, 1, 2, \ldots, 18\}$ the full set of reachable state while following the game rules? For which values of $x \in \mathcal{X}$ do you already explicilty know the values of $V_n(x)$, for every $n \geq 0$? (i.e. boundary conditions)

(b) Create a vector, V of length 19, to hold the values of $V_0(\cdot)$. Note that in R, the first element of a vector is called element 1, so V[i] will need to contain $V_0(i-1)$.

(c) Create another vector, W, this time holding the values of $V_1(\cdot)$. Only insert values from the boundary conditions. Values not known from the boundary conditions should be set equal to NA.

(d) Define a function of x, a, p and V_{T-1} which calculates the right-hand side (RHS) bracket of the optimality equation for the given values.

(e) Identify the valid choices $a \in \mathcal{A}_x$. Construct a loop over all valid (x, a) combinations (we only need $1 \leq x \leq 9$, and I suggest an outer loop over x, and inner loop over a). Test it by printing all (x, a) pairs.

(f) Inside the loop, add a step to call the function defined in (d) for each $a \in \mathcal{A}_x$ in order to find the best choice a at each x. Then for each $x \in \{1, 2, \ldots, 9\}$ store the best choice of a in some vector you create.

(g) In your loop above, use the the vector W to store the values of $V_T(x)$ calculated as the best value of the RHS of the optimality equation when given x and V_{T-1}. (You can use V to store V_{T-1}, and W for V_T.)

(h) Copy your code above inside the function shell described in the accompanying purple block from the notes (see previous page) and ...

Part II: Use your value function function to answer questions... most importantly comment on the answers!

- Find $V_1(x)$, $V_2(x)$ and $V_3(x)$ for $x \in \{1, 2, \ldots, 9\}$, when $p = 0.6$. Also finding the optimal first initial actions.
- How does $V_{1000}(5)$ vary for $p \in \{\frac{1}{10}, \frac{1}{3}, \frac{1}{2}, \frac{3}{4}\}$?
- What is the initial optimal betting action for each state when $p = 0.4$ and $T = 20$?
- How do $V_{1000}(x)$ and $V_{999}(x)$ compare?
- Feel free to answer other questions you pose yourself too!