```julia
using PlutoUI
```

```
parse_board_line (generic function with 1 method)
```

```
parse_file (generic function with 1 method)
```

Problem 1

```
isbingo (generic function with 1 method)
```

```julia
function isbingo(board)
    rows, cols = size(board)
    for r in 1:rows
        if (all(board[r,:] .== -1))
            return true
        end
    end

    for c in 1:cols
        if (all(board[:, c] .== -1))
            return true
        end
    end

    return false
end

```

```
simulate_bingo_games! (generic function with 1 method)
```

```julia
function simulate_bingo_games!(nums, boards)
    for num in nums
        (boards
            .|> b -> b[b .== num] .= -1)
        won_bingo = findall(isbingo, boards)

        if (length(won_bingo) >= 1)
            return num, boards[won_bingo][1]
        end
    end

end
```

```
50008
```

```
    0.004140 seconds (102.35 k allocations: 7.396 MiB)
```

```julia
 open("./Day4/prob_input.txt") do io

     with_terminal() do
        lines = [line for line in eachline(io)]
        nums, boards = parse_file(lines)
        @time num, winning_board = simulate_bingo_games!(nums, boards)
        winning_board[winning_board .== -1] .= 0
        num * sum(vcat(winning_board...))
     end
end
```

Problem 2

```
simulate_bingo_games_to_end! (generic function with 1 method)
```

```julia
function simulate_bingo_games_to_end!(nums, boards)
    remaining_boards = length(boards)
    for num in nums
        (boards
            .|> b -> b[b .== num] .= -1)
        won_bingo = findall(isbingo, boards)
        remaining_boards -= length(won_bingo)

        updated_boards = filter(bi -> bi ∉ won_bingo, eachindex(boards))

        if (remaining_boards == 0)
            return num, boards[won_bingo][1]
        end

        boards = boards[updated_boards]
    end

end
```

```
17408
```

```
  0.016393 seconds (199.92 k allocations: 14.592 MiB, 44.36% gc time)
```

```julia
 open("./Day4/prob_input.txt") do io

     with_terminal() do
        lines = [line for line in eachline(io)]
        nums, boards = parse_file(lines)
        @time num, winning_board = simulate_bingo_games_to_end!(nums, boards)

        winning_board[winning_board .== -1] .= 0
        num * sum(vcat(winning_board...))
     end
end
```