```julia
using PlutoUI
```

```julia
using BenchmarkTools
```

Problem 1

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

```julia
function parse_submarine_telemetry(io::IO)
    bitCount = zeros(Int, 12)
    length = 0
    for line in eachline(io)
        parsed_bits = collect.(line) |> x -> parse.(Int8, x)
        bitCount .+= parsed_bits
        length += 1
    end

    gamma = (map(x -> x > (length-x) ? 1 : 0, bitCount)
                |> join
                |> b -> parse(Int, b, base=2))

    epsilon = (map(x -> x > (length-x) ? 0 : 1, bitCount)
                |> join
                |> b -> parse(Int, b, base=2))

    return (gamma, epsilon)
end
```

```
3429254
```

```
    0.000499 seconds (3.56 k allocations: 261.156 KiB)
```

```julia
 open("./Day3/prob_1_input.txt") do io

     with_terminal() do
        @time (gamma, epsilon) = parse_submarine_telemetry(io)
        gamma*epsilon
    end
end
```

Problem 2

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

```
function isone(x) x == 1 end
```

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

```julia
function parse_submarine_vitals(bin_arr::Vector{Vector{Int8}})
    length_binaries = length(bin_arr)
    BITS = length(bin_arr[1])

    onesIndex = [[] for _ in 1:BITS]
    get_ones_indexes = x -> findall(isone, x)

    for (index, bits) in enumerate(bin_arr)
        push!.(onesIndex[get_ones_indexes(bits)], index)
    end

    oxygen = 1:length_binaries
    co2 = 1:length_binaries

    for index in 1:BITS
        in_ones_index(val) = val ∈ onesIndex[index]
        in_zeros_index(val) = val ∉ onesIndex[index]

        if (length(oxygen) > 1)
            oxygenBitCountOnes = filter(in_ones_index, oxygen) |> length
            oxygenBitCountZeros = length(oxygen) - oxygenBitCountOnes
            oxygen = filter(
                oxygenBitCountOnes >= oxygenBitCountZeros
                ? in_ones_index
                : in_zeros_index,
                oxygen)
        end

        if (length(co2) > 1)
            co2BitCountOnes = filter(in_ones_index, co2) |> length
            co2BitCountZeros = length(co2) - co2BitCountOnes
            co2 = filter(
                co2BitCountZeros <= co2BitCountOnes
                ? in_zeros_index
                : in_ones_index,
                co2)
```

```
BenchmarkTools.Trial: 57 samples with 1 evaluation.
 Range (min … max):  84.127 ms … 102.718 ms  ┊ GC (min … max): 0.00% … 0.00%
 Time  (median):     86.673 ms               ┊ GC (median):    1.89%
 Time  (mean ± σ):   87.761 ms ±   3.283 ms  ┊ GC (mean ± σ):  1.26% ± 0.98%
```

```
  84.1 ms         Histogram: frequency by time        93.7 ms <

 Memory estimate: 27.81 MiB, allocs estimate: 1755412.
```

```julia
@benchmark open("./Day3/prob_1_input.txt") do io
    with_terminal() do
        @time bin_arr = [collect.(line) |> x-> parse.(Int8, x) for line in
eachline(io)]

        @time (oxygen, co2) = parse_submarine_vitals(bin_arr)
        oxygen * co2
    end

end
```

```
5410338
```

```julia
5410338
```

Problem 2 *(Optimized)*

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

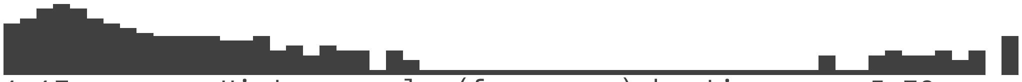
```julia
function parse_submarine_vitals_optim(bin_arr::Vector{Vector{Int8}})
    mx_bin_arr = hcat(bin_arr...)
    length_bin_arr = length(bin_arr)
    bits = length(bin_arr[1])
    isone_with_index(tup) = isone(first(tup))

    function find_val(f)
        vitalArr = 1:length_bin_arr
        for bit_num in 1:bits
            if (length(vitalArr) > 1)
                correspondingVals = zip(mx_bin_arr[bit_num, vitalArr], vitalArr) |>
collect
                indexesWithOnes = filter(isone_with_index, correspondingVals) .|>
last
                bitCountOnes = length(indexesWithOnes)
                bitCountZeros = length(vitalArr) - bitCountOnes
                vitalArr = filter(
                    f(indexesWithOnes, bitCountOnes, bitCountZeros),
                    vitalArr)
            else
                break
            end
        end

        return bin_arr[vitalArr[1]] |> join |> b -> parse(Int, b, base=2)
    end

    oxygen = find_val(
        (indexesWithOnes, bitCountOnes, bitCountZeros) ->
            bitCountOnes >= bitCountZeros
                ? b -> b ∈ indexesWithOnes
                : b -> b ∉ indexesWithOnes)
    co2 = find_val(
        (indexesWithOnes, bitCountOnes, bitCountZeros) ->
            bitCountZeros <= bitCountOnes
                ? b -> b ∉ indexesWithOnes
                : b -> b ∈ indexesWithOnes)
```

```
BenchmarkTools.Trial: 3247 samples with 1 evaluation.
 Range (min … max):  1.175 ms …   6.797 ms  ┊ GC (min … max): 0.00% … 68.12%
 Time  (median):     1.429 ms               ┊ GC (median):    0.00%
 Time  (mean ± σ):   1.530 ms ± 537.517 μs  ┊ GC (mean ± σ):  3.95% ±  8.68%
```



```
  1.17 ms      Histogram: log(frequency) by time      5.32 ms <

 Memory estimate: 729.08 KiB, allocs estimate: 4460.
```

```
@benchmark open("./Day3/prob_1_input.txt") do io
    with_terminal() do
        @time bin_arr = [collect.(line) |> x-> parse.(Int8, x) for line in
eachline(io)]

        @time (oxygen, co2) = parse_submarine_vitals_optim(bin_arr)
        #oxygen * co2
    end

end
```