```
using PlutoUI
```

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

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

```
function parse_fold_line(line)
    axis, point = match(r"^fold along ([x|y])=(\d+)", line).captures

    # We don't do +1 here as the point of fold isn't included in that grid
    return (axis == "x" ? FoldVertical : FoldHorizontal, parse(Int16, point))
end
```

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

```
function parse_file(io::IO)
    is_fold_line(line) = startswith(line, "fold")

    dot_coordinates = []
    fold_instructions = []

    for line in eachline(io)
        if is_fold_line(line) == true
            push!(fold_instructions, parse_fold_line(line))
        elseif !isempty(line)
            push!(dot_coordinates, parse_dot_line(line))
        end
    end

    return (dot_coordinates, fold_instructions)
end
```

```
@enum FoldType FoldHorizontal FoldVertical
```

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

```
function fold_paper(fold_type, position, grid)
    total_rows, total_cols = fold_type == FoldVertical ?
        (size(grid)[1], position) :
        (position, size(grid)[2])
    fold_dimension = fold_type == FoldVertical ? 2 : 1

    lhs_grid = grid[1:total_rows, 1:total_cols]
    rhs_grid = reverse(grid, dims=fold_dimension)[1:total_rows, 1:total_cols]
    return lhs_grid .| rhs_grid
end
```

```
fold_grid (generic function with 2 methods)
```

```
function fold_grid(grid, fold_instructions, on_fold = nothing)
    running_grid = grid
    for (fold_type, fold_position) in fold_instructions
        updated_grid = fold_paper(fold_type, fold_position, running_grid)
        on_fold != nothing && on_fold((fold_type, fold_position), copy(running_grid), copy(updated_grid))
        running_grid = updated_grid
    end

    return count(x -> x==true, running_grid), running_grid
end
```

Problem 1

```
701
```

```
  0.031124 seconds (84.84 k allocations: 6.512 MiB, 89.87% compilation time)
```

```
with_terminal() do
    open("./Day13/prob_input.txt") do io
        dot_coordinates, fold_instructions = parse_file(io)
        total_rows, total_cols = maximum(x -> x[2], dot_coordinates), maximum(x -> x[1], dot_coordinates)

        # Create initial grid
        grid = zeros(Bool, total_rows, total_cols)
        grid[map(x -> CartesianIndex(x[2], x[1]), dot_coordinates)] .= true

        @time count, final_grid = fold_grid(grid, [fold_instructions[1]])
        count
    end
end
```

Problem 2

Create simulation GIF? ☐

```
md"Create simulation GIF? $(@bind create_gif CheckBox())"
```

```
  0.099297 seconds (2.08 M allocations: 36.472 MiB, 14.94% gc time, 29.06% compilation
```

```
with_terminal() do
    open("./Day13/prob_input.txt") do io
        dot_coordinates, fold_instructions = parse_file(io)
        total_rows, total_cols = maximum(x -> x[2], dot_coordinates), maximum(x -> x[1], dot_coordinates)

        # Create initial grid
        grid = zeros(Bool, total_rows, total_cols)
        grid[map(x -> CartesianIndex(x[2], x[1]), dot_coordinates)] .= true

        folding_history = []
        fold_num = 1
        function on_fold(fold_props, before, after)
            push!(folding_history, after)
            #write_grid_presentation_to_file("./Day13/intermediate_$(fold_num).txt", fold_props, before, after)
            fold_num += 1
        end
        @time _, final_grid = fold_grid(grid, fold_instructions, on_fold)

        create_gif && create_folding_gif(folding_history, total_rows, total_cols)
        create_code_img.(folding_history[end])
    end
end
```

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

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

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