Advent of Code 2020

Aaron Kaw

Day 04 - Passport Processing

You arrive at the airport only to realize that you grabbed your NorthPole Credentials instead of your passport. While these documents are extremely similar, North Pole Credentials aren't issued by a country and therefore aren't actually valid documentation for travel in most of the world.

It seems like you're not the only one having problems, though; a very long line has formed for the automatic passport scanners, and the delay could upset your travel itinerary.

Due to some questionable network security, you realize you might be able to solve both of these problems at the same time.

The automatic passport scanners are slow because they're having trouble **detecting which passports have all required fields**. The expected fields are as follows:

- `byr` (Birth Year)
- `iyr` (Issue Year)
- `eyr` (Expiration Year)
- `hgt` (Height)
- `hcl` (Hair Color)
- `ecl` (Eye Color)
- `pid` (Passport ID)
- `cid` (Country ID)

Passport data is validated in batch files (your puzzle input). Each passport is represented as a sequence of `key:value` pairs separated by spaces or newlines. Passports are separated by blank lines.

Here is an example batch file containing four passports:

```
ecl:gry pid:860033327 eyr:2020 hcl:#fffffd
byr:1937 iyr:2017 cid:147 hgt:183cm

iyr:2013 ecl:amb cid:350 eyr:2023 pid:028048884
hcl:#cfa07d byr:1929
```

```
hcl:#ae17e1 iyr:2013
eyr:2024
ecl:brn pid:760753108 byr:1931
hgt:179cm

hcl:#cfa07d eyr:2025 pid:166559648
iyr:2011 ecl:brn hgt:59in
```

The first passport is **valid** - all eight fields are present. The second passport is **invalid** - it is missing `hgt` (the Height field).

The third passport is interesting; the **only missing field** is `cid`, so it looks like data from the North Pole Credentials, not a a passport at all! Surely, nobody would mind if you made the system temporarily ignore missing `cid` fields. Treat this "passport" as **valid**.

The fourth passport is missing two fields, `cid` and `byr`. Missing `cid` is fine, but missing any other field is not, so this passport is **invalid**.

According to the above rules, your improved system would report `2` valid passports.

```
struct Passport
    byr::String
    iyr::String
    eyr::String
    hgt::String
    hcl::String
    ecl::String
    pid::String
    cid::String
end

pps = open("puzzle_input.txt") do file
    passports = Vector{Passport}(undef, 0)
    byr = ""
    iyr = ""
    eyr = ""
    hgt = ""
    hcl = ""
    ecl = ""
    pid = ""
    cid = ""
    b_last_pushed = false
    for line ∈ eachline(file)
        if length(line) == 0
            push!(passports, Passport(byr, iyr, eyr, hgt, hcl, ecl, pid, cid))
            byr = ""
            iyr = ""
            eyr = ""
            hgt = ""
            hcl = ""
            ecl = ""
            pid = ""
            cid = ""
            b_last_pushed = true
        else
            for info ∈ split(line, " ")
                icolon = findfirst(":", info)[1]
```

```
                field = info[begin:icolon-1]
                data = info[icolon+1:end]
                if field == "byr"
                    byr = data
                elseif field == "iyr"
                    iyr = data
                elseif field == "eyr"
                    eyr = data
                elseif field == "hgt"
                    hgt = data
                elseif field == "hcl"
                    hcl = data
                elseif field == "ecl"
                    ecl = data
                elseif field == "pid"
                    pid = data
                elseif field == "cid"
                    cid = data
                else
                    ErrorException("Unrecognized field.") |> throw
                end
            end
            b_last_pushed = false
        end
    end
    if !b_last_pushed
        push!(passports, Passport(byr, iyr, eyr, hgt, hcl, ecl, pid, cid))
    end
    return passports
end
```

```
276-element Array{Main.##WeaveSandBox#273.Passport,1}:
 Main.##WeaveSandBox#273.Passport("2029", "2015", "1992", "59cm", "#b6652a"
, "#7a0fa6", "9381688753", "219")
 Main.##WeaveSandBox#273.Passport("1968", "2018", "2026", "", "", "blu", "9
43614755", "335")
 Main.##WeaveSandBox#273.Passport("2029", "2025", "1944", "64cm", "#ceb3a1"
, "#07219a", "067285985", "281")
 Main.##WeaveSandBox#273.Passport("1970", "2016", "2026", "185cm", "#866857
", "gry", "269105457", "222")
 Main.##WeaveSandBox#273.Passport("1990", "2012", "", "163cm", "#b6652a", "
brn", "260043570", "275")
 Main.##WeaveSandBox#273.Passport("1950", "1930", "2039", "181cm", "#b6652a
", "#906548", "604983466", "")
 Main.##WeaveSandBox#273.Passport("2006", "2025", "1956", "141", "z", "#f2a
ffc", "#1c42cc", "327")
 Main.##WeaveSandBox#273.Passport("1939", "2020", "2026", "178cm", "#888785
", "oth", "595705064", "")
 Main.##WeaveSandBox#273.Passport("1980", "2016", "2020", "159cm", "#efcc98
", "brn", "139063139", "")
 Main.##WeaveSandBox#273.Passport("1997", "2011", "2022", "179cm", "#602927
", "brn", "646870519", "")
 ⋮
 ⋮@*(Main.(*@##WeaveSandBox#273.Passport("1973", "2010", "2022", "177cm", "#fffffd
", "oth", "324648387", "60")
 Main.##WeaveSandBox#273.Passport("1928", "2017", "2023", "73in", "#efcc98"
, "brn", "632056596", "")
 Main.##WeaveSandBox#273.Passport("1943", "2012", "2035", "180cm", "", "amb
", "155cm", "144")
 Main.##WeaveSandBox#273.Passport("1958", "2019", "2023", "172cm", "#6b5442
```

```
", "gry", "927492391", "92")
 Main.##WeaveSandBox#273.Passport("2026", "2020", "2034", "193in", "#b6652a
", "grn", "", "82")
 Main.##WeaveSandBox#273.Passport("2015", "1922", "2040", "101", "245cb3",
"lzr", "151cm", "136")
 Main.##WeaveSandBox#273.Passport("2025", "2028", "2029", "193in", "z", "gr
y", "9335153289", "308")
 Main.##WeaveSandBox#273.Passport("1922", "2014", "2030", "163cm", "#ceb3a1
", "blu", "147768826", "169")
 Main.##WeaveSandBox#273.Passport("2002", "2020", "2028", "188cm", "#c0946f
", "blu", "998185490", "165")
```

1 Part One

Count the number of **valid** passports - those that have all required fields. Treat `cid` as optional. **In your batch file, how many passports are valid?**

1.1 Solution

```
function valid_passport(pp::Passport)
    if pp.byr |> length == 0
        return false
    elseif pp.iyr |> length == 0
        return false
    elseif pp.eyr |> length == 0
        return false
    elseif pp.hgt |> length == 0
        return false
    elseif pp.hcl |> length == 0
        return false
    elseif pp.ecl |> length == 0
        return false
    elseif pp.pid |> length == 0
        return false
    else
        return true
    end
end

num_valid = valid_passport.(pps) |> sum
```

```
210
```

2 Part Two

The line is moving more quickly now, but you overhear airport security talking about how passports with invalid data are getting through. Better add some data validation, quick!

You can continue to ignore the `cid` field, but each other field has strict rules about what values are valid for automatic validation:

- byr (Birth Year) - four digits; at least 1920 and at most 2002.

- iyr (Issue Year) - four digits; at least 2010 and at most 2020.
- eyr (Expiration Year) - four digits; at least 2020 and at most 2030.
- hgt (Height) - a number followed by either cm or in:
- If cm, the number must be at least 150 and at most 193.
- If in, the number must be at least 59 and at most 76.
- hcl (Hair Color) - a # followed by exactly six characters 0-9 or a-f.
- ecl (Eye Color) - exactly one of: amb blu brn gry grn hzl oth.
- pid (Passport ID) - a nine-digit number, including leading zeroes.
- cid (Country ID) - ignored, missing or not.

Your job is to count the passports where all required fields are both **present** and **valid** according to the above rules. Here are some example values:

```
byr valid:   2002
byr invalid: 2003

hgt valid:   60in
hgt valid:   190cm
hgt invalid: 190in
hgt invalid: 190

hcl valid:   #123abc
hcl invalid: #123abz
hcl invalid: 123abc

ecl valid:   brn
ecl invalid: wat

pid valid:   000000001
pid invalid: 0123456789
```

Here are some invalid passports:

```
eyr:1972 cid:100
hcl:#18171d ecl:amb hgt:170 pid:186cm iyr:2018 byr:1926

iyr:2019
hcl:#602927 eyr:1967 hgt:170cm
ecl:grn pid:012533040 byr:1946

hcl:dab227 iyr:2012
ecl:brn hgt:182cm pid:021572410 eyr:2020 byr:1992 cid:277
```

```
hgt:59cm ecl:zzz
eyr:2038 hcl:74454a iyr:2023
pid:3556412378 byr:2007
```

Here are some valid passports:

```
pid:087499704 hgt:74in ecl:grn iyr:2012 eyr:2030 byr:1980
hcl:#623a2f

eyr:2029 ecl:blu cid:129 byr:1989
iyr:2014 pid:896056539 hcl:#a97842 hgt:165cm

hcl:#888785
hgt:164cm byr:2001 iyr:2015 cid:88
pid:545766238 ecl:hzl
eyr:2022

iyr:2010 hgt:158cm hcl:#b6652a ecl:blu byr:1944 eyr:2021 pid:093154719
```

Count the number of **valid** passports - those that have all required fields **and valid values**. Continue to treat `cid` as optional. **In your batch file, how many passports are valid?**

2.1 Solution

```
function is_valid_byr(byr::AbstractString)
    if length(byr) == 4 && 1920 ≤ tryparse(Int, byr) ≤ 2002
        return true
    else
        return false
    end
end

function is_valid_iyr(iyr::AbstractString)
    if length(iyr) == 4 && 2010 ≤ tryparse(Int, iyr) ≤ 2020
        return true
    else
        return false
    end
end

function is_valid_eyr(eyr::AbstractString)
    if length(eyr) == 4 && 2020 ≤ tryparse(Int, eyr) ≤ 2030
        return true
    else
        return false
    end
end

function is_valid_hgt(hgt::AbstractString)
    if length(hgt) < 3
        return false
    end
    units = hgt[end-1:end]
```

```
    value = tryparse(Int, hgt[1:end-2])
    if value |> isnothing
        return false
    end
    if units == "cm"
        if 150 ≤ value ≤ 193
            return true
        else
            return false
        end
    elseif units == "in"
        if 59 ≤ value ≤ 76
            return true
        else
            return false
        end
    else
        return false
    end
end

function isalphabetic(ch::AbstractChar)
    if 'a' ≤ ch ≤ 'z'
        return true
    elseif 'A' ≤ ch ≤ 'Z'
        return true
    else
        return false
    end
end

function is_valid_hcl(hcl::AbstractString)
    if length(hcl) < 2
        return false
    end
    if hcl[1] ≠ '#'
        return false
    end
    value = hcl[2:end]
    for ch ∈ value
        if !(isnumeric(ch) || isalphabetic(ch))
            return false
        end
    end
    return true
end

function is_valid_ecl(ecl::AbstractString)
    if ecl == "amb" || ecl == "blu" || ecl == "brn" || ecl == "gry" || ecl == "grn" ||
ecl == "hzl" || ecl == "oth"
        return true
    else
        return false
    end
end

function is_valid_pid(pid::AbstractString)
    value = tryparse(Int, pid)
    if value |> isnothing
```

```
        return false
    end
    if length(pid) ≠ 9
        return false
    end
    return true
end

function valid_passport(p::Passport)
    if !is_valid_byr(p.byr)
        return false
    elseif !is_valid_iyr(p.iyr)
        return false
    elseif !is_valid_eyr(p.eyr)
        return false
    elseif !is_valid_hgt(p.hgt)
        return false
    elseif !is_valid_hcl(p.hcl)
        return false
    elseif !is_valid_ecl(p.ecl)
        return false
    elseif !is_valid_pid(p.pid)
        return false
    else
        return true
    end
end

num_valid = valid_passport.(pps) |> sum
```

```
131
```