A Culinary Journey through Advent of Code

by u/WilkoTom

Advent of Cookery

Introduction

Now in its ninth year, Advent Of Code has become a highlight of each December. Christmas isn't really on its way until I'm late for work because I've been desperate to fix a bug in a program that's trying to parse some particularly esoteric format.

This year's Community Fun theme is *Allez Cuisine!*; I couldn't think of a better way to celebrate this than to pair my answer to each day's puzzle with a recipe.

DAY 1

Trebuchet?!

Day 1
Things are even worse than I thought...

Well, this is… different.

Almost every year so far the first challenge has consisted of adding up a bunch of numbers. And thinking about it, that's true of today as well. Problem is the parsing for these is not just cast to int but a little more involved. To my shame, this is the first time ever I've submitted a wrong answer for a day 1 puzzle. Twone and eighthree be damned.

Anyway, today's Upping The Ante challenge calls for the use of no more than two variables. The result is a Python script that made me feel ill after writing it, and made me think of Kraft Mac'n'Cheese, a dish that nobody in their right mind would choose to eat having looked at the ingredients.

```
print(sum
(int(l.strip("abcdefghijklmnopqrstuvwxyzABCDEFGHI
JKLMNOPQRSTUVWXYZ\n")[0] +
l.strip("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ\n")[-1]) for l in open(".aochelpers/
2023/1").readlines()),sum (int(l.replace("one",
"o1e").replace("two", "t2o").replace("three",
"t3e").replace("four", "f4r").replace("five",
"f5e").replace("six", "s6x").replace("seven",
"s7n").replace("eight", "e8t").replace("nine",
"n9e").strip("abcdefghijklmnopqrstuvwxyzABCDEFGHI
JKLMNOPQRSTUVWXYZ\n")[0] + l.replace("one",
"o1e").replace("two", "t2o").replace("three",
"t3e").replace("four", "f4r").replace("five",
"f5e").replace("six", "s6x").replace("seven",
"s7n").replace("eight", "e8t").replace("nine",
"n9e").strip("abcdefghijklmnopqrstuvwxyzABCDEFGHI
JKLMNOPQRSTUVWXYZ\n")[-1]) for l in
open(".aochelpers/2023/1").readlines()))
```

Mushroom and Pancetta Macaroni Cheese

Rather than the horrors that come out of a box containing some sort of powdered cheese analogue, here's a version of the dish I'll happily eat.

Ingredient	Amount	
Dry Macaroni	400	g
Diced Pancetta	100	g
Baby chestnut mushrooms, chopped	200	g
Milk	400	ml
Breadcrumbs, fresh	100	g
Cheddar cheese, grated	175	g
Mozarella, drained and diced	100	g
Parmaggio Reggiano, grated	60	g
Butter	85	G
Mustard Powder	1	tsp

Method

1. Bring a large pan of lightly salted water to the boil over a high heat. Add the macaroni and boil for 2 minutes less than specified on the packet. Drain well and set aside, shaking off any excess water.

2. Meanwhile, preheat the oven to 200°C (400°F/Gas 6) and grease an ovenproof serving dish. Melt 25g (scant 1oz) of the butter in a small pan. Add the breadcrumbs, stir, then remove the pan from the heat and set aside.

3. Place a frying pan over a medium heat, and add the pancetta. Once it has started to cook, add the mushrooms

4. Melt the remaining butter in a large saucepan over a medium heat. Sprinkle over the flour, then stir for 30 seconds. Stir in the mustard powder and nutmeg, then remove the pan from the heat and slowly whisk in the milk. Return the pan to the heat and bring the mixture to the boil, whisking, for 2–3 minutes, or until the sauce thickens. Remove from the heat. Stir in the Cheddar cheese, until melted and smooth, then add the macaroni and mozzarella and stir together.

5. Transfer the mixture to the prepared dish and smooth the surface. Toss the breadcrumbs with the Parmaggiano Regiano and sprinkle over the top. Place the in the oven and bake for 25 minutes, or until heated through and golden brown on top. Leave to stand for 2 minutes, then serve straight from the dish.

DAY 2

Cube Conundrum

Day 2
Healthy, tasty travels

This is more like it; an easy to understand specification followed by a puzzle that's not too taxing. And because the theme of the year is cuisine, it occurred to me that the cubes the Elf was producing from his bag looked like they might be part of a Greek Salad; green for cucumber, red for tomato and blue for feta (well, there's a lot of blue on the Greek Flag).

A simple salad for a simple solution, with a few kitchen-themed identifiers thrown into my code for good measure.

```
fn chop_salad(ingredients: &str) -> Salad {
    let mut greek_salad = Salad{ tomato: 0, cucumber: 0, feta: 0 };
    for ingredient in ingredients.split(", ") {
        let mut colours = ingredient.split(' ');
        let count=colours.next().unwrap().parse::<i32>().unwrap();
        match colours.next() {
            Some("red") => {
                greek_salad.tomato += count
            },
            Some("blue") => {
                greek_salad.feta += count
            },
            Some("green") => {
                greek_salad.cucumber += count
            },
            Some(_) | None => unimplemented!(),
        }
    }
    greek_salad
}
```

u/WilkoTom's Greek(ish) Salad

This salad bears as much resemblance to a real Greek Salad as Advent of Code does to a real programming job. Nevertheless, it disappears very quickly at family barbecues.

Ingredient	Amount	
Gem Lettuce	1	Head
Feta Cheese	150	g
Baby Plum tomatoes	300	g
Cucumber	1	
Black Olives, stoned	100	g
Olive Oil for drizzling		

Method

1. Chop cheese, cucumber and tomatoes into cubes, approximately 3/4 inch on a side
2. Wash and shred the lettuce
3. Place all ingredients into a large salad bowl, drizzle with oil to taste
4. Eat immediately.

DAY 3

Gear Ratios

Day 3
Grinding My Gears

…and there we go, back to a level of difficulty I'd previously associated with days 10-15. Well, probably not, but the combination of "Sunday Puzzle", which have historically been on the harder and of the scale and an early day meant that I had to think a bit more than I'd been hoping for.

On the bright side, I didn't see anyone else solving the problem the same way I had; the first time I encountered a value touching a multiplier operator, I tagged it with that value. The next time I found the same operator, I knew both components of the component's value.

The theme for the day is spam, which immediately made me think of the Monty Python spam song. I think my code captures the nature of the song perfectly.

```
fn spamandchips(data: &str)-> (SpamSpamSpamSpam, SpamSpamSpamSpam)
{
    let mut spam = spam_spam(data);
    let mut spamspam = WonderfulSpam{ x: 0, y: 0 };
    for (spam, _) in spam.iter() {
        spamspam.x = spamspam.x.max(spam.x);
        spamspam.y = spamspam.y.max(spam.y);
    }
    let mut spamandchips: SpamSpamSpamSpam = 0;
    let mut spamspamspam: SpamSpamSpamSpam = 0;

    for beautifulspam in 0..=spamspam.y {
        let mut spamspameggandchips = 0;
        let mut spamspamspameggchipsandspam = false;
        let mut spamspamspamspamspamspamspamspamwonderfulspam = No
        for lovelyspam in 0..=spamspam.x {
            match spam.get(&WonderfulSpam{x: lovelyspam,y: beautif
                Some(Spam::Spam(spamspamspamspam)) => {
                    spamspameggandchips *= 10;
                    spamspameggandchips += spamspamspamspam;
                    if !spamspamspameggchipsandspam {
                        for lovelyspam in (WonderfulSpam{x: lovely
                            match spam.get(lovelyspam) {
                            Some(Spam::SpamSpam(_)) => {
                                spamspamspameggchipsandspam = true
                                break;
                            }
                            Some(Spam::SpamSpamSpam(_)) => {
                                spamspamspameggchipsandspam = true
                            _ => {},
                        }
                    }
                }
            },
spamspamspamspamspamspamspamspamwonderfulspam = Some(*lovelyspam);
```

Spam Musubi Fritters

Spam Fritters were a commonly served dish in my school canteen, even then being something of an anachronism. They were cheap to make, and because they were deep-fried, they were also delicious. Decades later I was introduced to Spam Musubi on a business trip. The experimental chef in me couldn't help combining the two in the Scottish fashion- Frankenfood at its best (or worst).

Ingredient	Amount	
Spam	1	Can
Self-Raising Flour	100g	g
Sushi Rice	180	g
Nori seaweed sheets		
Sesame oil	1	tbsp
Salt	1/2	tsp
Sunflower oil for deep-frying	300	ml

Method

1. Put rice and 950ml of water in a saucepan. Bring to the boil, stirring occasionally, them reduce the heat as low as possible and simmer for 18-20 minutes until all the water has been absorbed. Remove from heat and stand for 10 minutes.

2. Slice the spam into 1/2 inch pieces

3. Press the rice into rectangular shapes, approximately the size and thickness of a piece of spam and place a slice of spam on top, wrapping a 1cm wide piece of nori around.

4. Mix flour, sesame oil and salt together and add 100 ml water to form a batter. Whisk to the consistency of single cream, and leave to stand for 10 minutes.

5. Half-fill a heavy-bottomed wok or saucepan with sunflower oil, and heat until the oil reaches 180 degrees celsius (360 Fahrenheit)

6. Take each piece of Spam Musubi, and coat it in the batter, before placing in the oil. Fry until golden-brown, then drain on kitchen paper.

7. Eat with chopsticks, accompanied by sweet chilli jam.

DAY 4

Scratchcards

Day 4
An Itch To Scratch?

Fun fact: The most I've ever won on the lottery is around £200. It sucked; any other week, that number of matches against my ticket would have put my winnings in the thousands.

I can't say I've ever won anything on a scratchcard, but then I've almost never played them either.

Once again, the difficulty level takes a hard swerve, giving us a conundrum that immediately reminded me of 2021's lanternfish puzzle. The solution was short enough I could turn it into a program that fit on a punchcard, with extra space to make the whole thing look a bit like an after-eight mint.

```
use aochelpers::get_daily_input; fn main(){println!("{:?}",bp(&get_daily_input(4
,2023).unwrap()))}fn ps( l:&str)->u32 {let mut p=l.split(" | ");let mc=p.next().
unwrap().split(' ').filter_map(|nm:&str| nm.parse::<i32>().ok()).collect::<Vec<_
>>( ); let /*                                                    */  w=p.next ()
.unwrap()/*   Superior-Quality Digestive Mint. Eases Indigestion.  */.split(' ')
.filter_map/*                                                    */(|nm|nm.parse
::<i32>().ok()).collect::<Vec<_>>();mc.iter().filter(|n|w.contains(n)).count()as
u32}fn bp(d:&str)->(i32,i32){let mut r=0;let c=d.split('\n').collect::<Vec<_>>()
;let mut z=vec![1;c.len()];for(i,x)in c.iter().enumerate(){let s=ps(x);if s>0{ r
+=i32::pow(2,s-1);for x in i+1..=(i+s as usize){z[x]+=z[i]}}}(r,z.iter().sum())}
```

Mojito

It's hard to make a recipe that features the humble IBM punchcard as an ingredient, though I furnished my code into one in the form of an after-dinner mint.

After-dinner mints are both boring and time-consuming to make, though; why not have a Mojito instead?

Ingredient	Amount	
Juice of one lime		
Mint Leaves, fresh	1	Small handful
Granulated sugar	1	tsp
White Run	60	ml
Soda Water		To taste
Crushed Ice		

Method

1. Place mint leaves, lime juice and sugar in a jug and muddle, ensuring to bruise the leaves.
2. Pour mudddled mixture into a hi-ball, adding a scoop of ice
3. Pour over the rum, stirring with a cocktail spoon, before topping up with soda water
4. Garnish with an additional sprig of mint and serve.

DAY 5

If you give a seed a fertiliser

Day 5
Let's make you fly up into the sky...

The theme for today is "Explain like I'm Five"; I did so by taking inspiration from a favourite of my own children when they were younger: *The Gruffalo, by* rewriting the Advent of Code story so far into a set of rhyming couplets.

In terms of a solution, I'm sorry to say that the brute-force approach I left running for part 2 completed before I'd even written a single line of code. Sometimes it pays to be smart; this wasn't one of those occasions.

```
fn main() -> Result<(), Box<dyn Error>>{
    let data = get_daily_input(05,2023)?;
    let almanac = parse_almanac(&data);
    println!("Part 1: {}", part1(&almanac));
    println!("Part 2: {}", part2(&almanac));
    Ok(())
}
fn part1(almanac: &Almanac) -> i64 {
    almanac.seed_list.iter().map(|s| grow_seed(*s,
almanac)).min().unwrap_or(0)
}

fn part2(almanac: &Almanac) -> i64 {
    // TODO: Refactor brute force
    let mut values = almanac.seed_list.iter();
    let mut result = i64::MAX;

    while let Some(start) = values.next() {
        let count = values.next().unwrap();
        result = result.min(grow_seeds(*start, *count,
almanac));
    }
    result
}

fn grow_seeds(start:i64, range: i64, almanac: &Almanac)
    let mut result = i64::MAX;
    for i in start..start+range {
```

Rice Krispie Cakes

Once you've tried to explain like I'm five, then why not try to cook like you're five? An easy recipe beloved of children everywhere. I much enjoyed cooking this recipe when I was 5, and the five-year-old in you will love eating them.

Ingredient	Amount	
Milk Chocolate	100	g
Rice Krispies	40	g
Fairy Cake Cases	8	

Method

1. Melt the chocolate in a glass mixing bowl. Most recipes call to do this by putting the bowl over a pan of hot water, but when I was five, I just used the microwave, under adult supervision, and you should too. Take care not to overheat the chocolate or it will become grainy.
2. Mix the Rice Krispies in with the melted chocolate, stirring until they are evenly coated.
3. Distribute the mixture evenly between the fairy cake cases, then refrigerate until set.
4. Enjoy, while reading *The Beano*, or perhaps watching *Bluey*.

DAY 6

Wait For It...

Day 6
Christmas was better in the ‘80s

Sometimes it seems like everything I’ve ever learned about technology is obsolete, sometimes even before I’ve heard of the object of study. So it was with great pleasure that today’s theme gave me the opportunity to use a tool I haven’t touched in several decades: BBC BASIC.

The challenge itself, easily solved with a quadratic formula on a modern machine, provided something of a challenge for a computer which supported only 16-bit numbers, and took it upon itself to lossily convert to floating point because nobody needs precision for big numbers.

Fortunately, there are libraries available for unlimited precision arithmetic or else I’d have given up on part 2 without a second thought. Thanks again to u/instantiator for sharing his work.

```
10 RACECOUNT = 3
20 DIM RACES(RACECOUNT-1,2)
30 FOR I = 0 TO 1
40 FOR J = 0 TO RACECOUNT-1
50 READ RACES(J,I)
60 NEXT J,I
70 RESULT = 1
80 FOR I = 0 TO RACECOUNT-1
90 COUNTER = 0
100 FOR J = 0 TO RACES(I,0)
110 IF (RACES(I,0) -J) * J > RACES(I,1) THEN COUNTER =
120 COUNTER + 1
130 NEXT J
140 RESULT = RESULT * COUNTER
150 NEXT I
160 PRINT "Part 1 "; RESULT
170
180 PROCinit
190 DIM FINALRACE$(2)
200 FOR F = 0 TO 1
210 PROCset(0,0)
220 FOR I = 0 TO RACECOUNT-1
230 L = LOG(RACES(I,F))
240 COUNTER = 0
250 REPEAT
260 PROCset(1,10)
270 PROCmul
280 COUNTER = COUNTER + 1
290 UNTIL COUNTER > L
```

Pork in Raisin Sauce

An early 1980s recipe to go with the early 1980s programming language used in implementing the solution.

Serve with plain white rice for that true 1980s culinary experience.

Ingredient	Amount	
Diced Pork shoulder	500	g
Butter	50	g
Garlic, crushed	1	clove
Plain Flour	40	g
Chicken Stock	425	ml
Honey	1	tbsp
Vinegar	2	tbsp
English mustard, made according to instructions	1	tsp
Raisins	25	g
Tomato Ketchup	2	tbsp
Sunflower Oil	2	Tbsp

Method

1. Pre-heat the oven to 180 degrees celsius.
2. Heat 25g of the butter in a frying pan over a medium heat, and add the pork shoulder. Fry until browned on each side, then remove from the pan and place in an ovenproof dish with a lid.
3. Add the remaining butter to the pan and allow to melt.
4. Add the garlic to the pan and fry until softened. Reduce the heat to low, and sprinkle in the flour.
5. Remove the pan from the heat and stir in the stock, a little at a time, before returning to the heat and bringing to the boil.
6. Turn off the heat and add the remaining ingredients, stirring until mixed well
7. Pour the contents of the saucepan over the pork, and place in the oven, baking for 1 hour.

DAY 7

Camel Cards

Day 7
Know when to hold ‘em...

I haven’t played a lot of cards, though I know a few people who have been partial to the odd trip to the Empire Poker Room when they’re feeling a little flush.I know the rules, though; there was a time when Late Night Poker on Channel 4 was compulsory viewing after a trip to the pub, and I’ve been known to hold my own when playing an equally badly-qualified group of players.

Following day 5’s childrens’ poetry theme, today I took a crack at a sonnet, and had a reasonable degree of success. Finding a food pairing for Shakespearean literature styles, though, is beyond me, so today’s recipe is instead inspired by the game we’re playing with the mysterious Elf clothed in white.

```
fn main() -> Result<(), Box<dyn Error>>{
    let hands = build_bids(&get_daily_input(7,2023)?);
    println!("Part 1: {}", play_poker(&hands, |a,b| comp
    println!("Part 2: {}", play_poker(&hands, |a,b| comp
Ok(())
}
```

```
fn play_poker(hands: &HashMap<String, i64>,
compare_hands: fn(&&String, &&String) -> Ordering ) ->
    let mut hand_list: Vec<&String> =
hands.keys().collect::<Vec<_>>();
    hand_list.sort_unstable_by(compare_hands);
    let mut total = 0;
    for (pos, hand) in hand_list.iter().enumerate() {
        total += (pos as i64 +1 ) * hands.get(*hand).unw
    }
    total
}

fn build_bids(data: &str) -> HashMap<String, i64> {
    let mut hands = HashMap::new();
    for hand in data.split('\n'){
        let mut tokens = hand.split(' ');
        let cards = tokens.next().unwrap();
        let bid =
tokens.next().unwrap().parse::<i64>().unwrap();
        hands.insert(cards.to_string(), bid);
    }
    hands
```

Chilli Con Carne

Any game night, whether you're playing Camel Cards, Rambunctious Recitation, or Cube Conundrum needs the right snack-based accompaniment. This recipe is just the thing you need to keep you going, even on those long evenings shuffling your Space Cards.

Ingredient	Amount	
Lean Minced Beef (5% fat)	500	g
Red Pepper, diced	1	
Onion, large, diced	1	
Garlic, finely chopped	2	cloves
Canned Chopped Tomatoes	440	g (one tin)
Canned Kidney Beans, drained	400	g (one tin)
Dark Chocolate	10	G
Tomato Puree	2	tbsp
Beef Stock	300	ml
Hot chilli powder	1	heaped tsp
Smoked Paprika	1	tsp
Cumin	1	tsp
Dried Marjoram	1/2	tsp

Method

1. Over a medium heat, fry the onions until they start to soften, stirring frequently.

2. Add the garlic, pepper, paprika, cumin and chilli powder to the pan. Continue to cook for a further five minutes.

3. Add the minced beef, continuing to stir until the meat has browned.

4. Add tomatoes, chocolate, marjoram and salt and pepper to taste. Cover the pan, reduce to a low heat and simmer for twenty minutes, stirring occasionally.

5. Add the kidney beans and cook for a further 10 minutes

6. Leave to stand for a further 10 minutes, then serve with sour cream and tortilla chips.

DAY 8

Haunted Wasteland

Day 8
Who ya gonna call?

Not me, certainly, if you're lost in a sandstorm in the desert with no hope of rescue and only a bunch of maps intended for ghosts to give you any hope of getting out.

The theme for the day is International Ingredients. I had a lot of fun translating my program into Japanese, translating it back and forward using various engines to make sure everything still made sense. Sadly, the developers of Rust put an end to my original idea before I'd even started - variable names using the International language of emoji.

```
fn main() -> 結果{
    let (方向のセット, 地図) = データを理解する(&get_daily_inp
    println!("前編 {}", 指示に従う(&方向のセット, &地図, "AAA"
    println!("後半: {}", 精神を導(&方向のセット, &地図));
    Ok(())
}

fn 指示に従う(方向: &str, 地図: &ハッシュマップ, 出発点: &str)
-> usize {
    let mut 実行された手順 = 0;
    let mut 私の場所 = 出発点;
    while !私の場所.ends_with('Z') {
        if 方向.chars().nth(実行された手順 % 方向.len()).un
            私の場所 = &地図.get(私の場所).unwrap().右;
        } else {
            私の場所 = &地図.get(私の場所).unwrap().左;
        }
        実行された手順 += 1
    }
    実行された手順
}

fn 精神を導(方向: &str, 地図: &ハッシュマップ) -> usize {
    地図.keys().filter(|バツ| バツ.ends_with('A')).map(|バ
}

fn データを理解する(資料: &str) -> (糸, ハッシュマップ) {
```

Chicago-Style Pizza

Today's ghostly theme brought to mind the *Ghostbusters* movies, so that's where today's dish takes us.

A scene in *Ghostbusters II* establishes that, despite being New Yorkers, the original Ghostbusters' favourite style of pizza is Chicago-style, so here we are…

Ingredient	Amount	
Plain flour	200	g
Polenta	30	g
Butter, melted	30	g
Butter, softened	90	g
Caster Sugar	1/2	tbsp
Lukewarm water	150	ml
Salt	1	tsp
Onion, grated	1/2	Small
Dried Oregano	2	tsp
Garlic, minced	1 1/2	Cloves
Canned Chopped Tomatoes	440	g (1 tin)
Mozarella, grated	200	g
Parmaggiano Regiano, grated	20	g
Pepperoni, sliced	60	g
Olive Oil	1	tbsp

Method

1. Mix together flour, polenta, 1/2 teaspoon salt, sugar and yeast in a clean dry mixing bowl. Add the warm water and the melted butter, taking care to ensure that the butter is only just melted.

2. Stir for 5 minutes, or until the dough takes form and no longer sticks. Place the dough on a clean worktop and cover with the mixing bowl, leaving for at least 2 hour, or until the dough has doubled in size.

3. Lightly flour a work surface, then roll the dough out into a circle. Spread the remainder of the butter across it, then roll the dough lengthways with the buttered side inside, before reforming it into a ball. Return the dough to rise again for a further hour.

4. Roll the dough flat a second time, and use it to line a 9-inch spring form cake pan.

5. Bring a saucepan to a medium heat, then add the olive oil. Fry the onion with the salt and oregano for 5 minutes, before adding the garlic, tomatoes and a pinch of sugar. Turn the heat down slightly and allow to simmer until reduced in volume by about half. Remove from heat.

6. Pre-heat the oven to 220C

7. Add the mozzarella to the cake pans, and then top with the pepperoni. Finally pour over the sauce, and generously sprinkle over the parmesan cheese.

8. Bake pizza for 20-30 minutes, until the crust is golden brown. Allow to stand for 10 minutes before serving.

DAY 9

Mirage Maintenance

Day 9
Sand? It gets everywhere...

Welcome to the Oasis. Or is it the O.A.S.I.S.? Wherever we are, it's a welcome reprieve from travels through the desert. And today's puzzle is, curiously enough, a welcome reprieve from a slog of difficulty. In fact, it's so straightforward that I'm left wondering when the other shoe is going to drop.

Today's theme is Advertising. In the spirit of the best advertisers everywhere, I used a number of quotes, some completely out of context to persuade my audience to try my newest dish, detangled spaghetti.

```
use std::error::Error;
use aochelpers::get_daily_input;

fn main() -> Result<(), Box<dyn Error>>{
    let dataset = get_daily_input(9,2023)?.split('\n')
        .map(|l| l.split(' ')
            .filter_map(|x| x.parse::<i64>().ok())
            .collect::<Vec<_>>())
        .collect::<Vec<_>>();
    println!("Part 1 {}", dataset.iter().map(
        |l| extrapolate_last_value(l)).sum::<i64>());
    println!("Part 2 {}", dataset.iter().map(
        |l| extrapolate_last_value(&l.iter().rev()
          .copied().collect::<Vec<_>>())).sum::<i64>());
    Ok(())
}

fn extrapolate_last_value(values: &[i64]) -> i64 {
    let differences =
        values[1..].iter().enumerate().map(
        |(i, x)| x - values[i]).collect::<Vec<_>>();
    if differences.iter().all(|x| x == &0) {
        *values.iter().last().unwrap()
    } else {
        values.iter().last().unwrap() +
           extrapolate_last_value(&differences)
    }
}
```

Pineapple Sorbet

The word “Oasis” brings to mind a tropical paradise. What could be more suited to a respite in the desert than this delicious, cooling dessert?

Ingredient	Amount	
Pineapple	1	
Sugar	120	g
Water	240	ml
Lemon Juice	1	tsp

Method

1. Dice the pineapple, and place in a blender with the sugar and water, blending until smooth.
2. Place mixture into a Tupperware container, seal, and freeze for 1-2 hours
3. Return to blender, and blend until smooth. Before returning to the freezer.
4. Freeze for at least 5 hours, then serve with fresh mint leaves.

DAY 10

Pipe Maze

Day 10
A Maze of Twisty Pipes, all alike?

The input for today's puzzle brings to mind Mine Cart Madness, a puzzle from five years before, the difference being that there's just one track and the initial task, for barely tenuous reasons, is to find half of its length. This was a fun one to think about, especially part 2, which, it turns out, it has at least three vastly different, but equally valid, approaches.

```
use std::error::Error;
use aochelpers::get_daily_input;

fn main() -> Result<(), Box<dyn Error>>{
    let dataset = get_daily_input(9,2023)?.split('\n')
        .map(|l| l.split(' ')
            .filter_map(|x| x.parse::<i64>().ok())
            .collect::<Vec<_>>())
        .collect::<Vec<_>>();
    println!("Part 1 {}", dataset.iter().map(
        |l| extrapolate_last_value(l)).sum::<i64>());
    println!("Part 2 {}", dataset.iter().map(
        |l| extrapolate_last_value(&l.iter().rev()
          .copied().collect::<Vec<_>>())).sum::<i64>());
    Ok(())
}

fn extrapolate_last_value(values: &[i64]) -> i64 {
    let differences =
        values[1..].iter().enumerate().map(
        |(i, x)| x - values[i]).collect::<Vec<_>>();
    if differences.iter().all(|x| x == &0) {
        *values.iter().last().unwrap()
    } else {
        values.iter().last().unwrap() +
           extrapolate_last_value(&differences)
    }
}
```

Session India Pale Ale

A maze of pipes makes me think of one particular type of place: a craft beer taproom, the kind of place that has twenty beers on tap on a bad day and whose bar staff can always find the right drink for you. This one has a lot of flavour and isn't too strong, though drinking the whole 20l this recipe makes at once is probably a very bad idea.

Ingredient	Amount	
Pale Malt	2800	g
Caramalt	630	g
Crystal Malt	200	g
Simcoe Hops	55	g
Citra Hops	85	g
Mosaic Hops	60	g
US-05 Ale yeast	11	G
Water	28	L
Sugar (for carbonation)	100	G

Method

1. Heat the water to 65C, remove from heat and add the malt, in a large muslin sack lining the pan. Stir until all the grains are separated.
2. Bring the temperature back to 62C and hold for 1 hour. Bring to 70 degrees then remove the malt, squeezing the bag and allowing as much liquid to drain back into the pan as possible.
3. Bring the pan to the boil and add 5g each of the Simcoe and Citra hops. Do not cover.
4. After 45 minutes, add a further 5 g each of Simcoe and Citra hops, and boil for a further 15 minutes.
5. Chill the liquid to room temperature as quickly as possible, then transfer the liquid into a sterilised fermenting vessel, add the yeast and seal.
6. Hold the fermentation vessel at 19 degrees for 2 days, then add the remaining hops.
7. Transfer to 500ml bottles, adding 2.5g of sugar to each before capping
8. Keep bottles at room temperature for a day or so before refrigerating.
9. Serve cold with spicy food.

DAY 11

Cosmic Expansion

Day 11
Beyond The Stars

Even when I read the description of part 1, I had a decent idea that part 2 would involve stars much further away, so I eschewed the obvious solution of inserting padding in the star field for double-counting rows and columns that were empty. That put me in a good place for part 2, where minimal tweaks were necessary.

What I didn't have time for was the challenge of further upping the ante. *One variable indeed!*

```
fn main() -> Result<(), Box<dyn Error>>{
    let data = get_daily_input(11,2023)?;
    let stars = generate_starfield(&data);
    println!("{}", star_distance(&stars, 2));
    println!("{}", star_distance(&stars, 1000000));
    Ok(())
}

fn star_distance(stars: &[Coordinate<i64>], empty_cost: i64) ->
i64 {
    let max_x = stars.iter().map(
        |c: &Coordinate<i64>| c.x).max().unwrap();
    let max_y = stars.iter().map(
        |c: &Coordinate<i64>| c.y).max().unwrap();

    let empty_columns = (0..max_x).filter(|x| stars.iter().all(
        |c| c.x != *x)).collect::<Vec<_>>();
    let empty_rows= (0..max_y).filter(|y| stars.iter().all(
        |c| c.y != *y)).collect::<Vec<_>>();
    let mut total = 0;
    for (idx, left_star) in stars.iter().enumerate() {
        for right_star in stars.iter().skip(idx+1){
            let empty_cols_count = empty_columns.iter().filter(
        |c| *c > &right_star.x.min(left_star.x) &&
            *c < &right_star.x.max(left_star.x)).count() as i64;
            let empty_rows_count = empty_rows.iter().filter(
        |c| *c > &right_star.y.min(left_star.y) &&
            *c < &right_star.y.max(left_star.y)).count() as i64;
            let distance=left_star.manhattan_distance(right_star)
         + ((empty_cols_count + empty_rows_count) *
           (empty_cost -1));
            total += distance
        }
    }
    total
}
```

Asteroid Cakes

Rock Cakes, really. Short of upping the ante by producing a space-themed single ingredient recipe ("Take freeze-dried ice cream out of packet. Serve"), this is the closest to a space-themed dish that I could think of that wasn't just a sweet treat with a themed decoration.

Ingredient	Amount	
Self-Raising Flour	225	g
Caster Sugar	75	g
Unsalted butter, cubed	125	g
Mixed Dried Fruit	150	g
Egg	1	
Milk	1	tbsp
Vanilla Extract	2	tsp
Baking Powder	1	Tsp

Method

1. Pre-heat the oven to 180C/ 160C fan. Line a baking tray with greaseproof paper.

2. Mix Flour, sugar and baking powder together in a bowl, add the butter and rub together until the mixture resembles fine breadcrumbs. Mix in the dried fruit.

3. In a separate bowl, whisk together the egg, milk and vanilla extract.

4. Add the liquid to the first bowl, and mix until the dough is thick and lumpy.

5. Spoon golf-ball sized lumps of dough onto the baking tray, leaving plenty of space for them to spread out.

6. Bake for 15-20 minutes, until golden brown, stand for 2 minutes, then place on a wire rack to cool.

DAY 12

Hot Springs

Day 12
Jack-In-The-Box

Just when you thought you were safe, up bounces Jack,powered by springs, out of his box. This seems to have been the first properly difficult problem this month. Dynamic programming and memoization are not my string suits, but I got there eventually. Of course, my first attempts took multiple seconds per line of input, so that was never going to work. I'm reminded of the elephant puzzle from last year; that's another one I'm never going to forget.

```
fn validate_springs(springs: &str,
          counts: &[usize],
          cache: &mut HashMap<(String,Vec<usize>), usize>)
 -> usize {

    let cache_key = (springs.to_string(), counts.to_vec());
    let result = if let Some(res) = cache.get(&cache_key) {
        *res
    } else if let Some(stripped) =
                        springs.strip_prefix('.') {
        validate_springs(stripped, counts, cache)
    } else if springs.starts_with('?') {
        let unknown_is_spring: String =
                        springs.replacen('?', "#", 1);
        validate_springs(&unknown_is_spring,
                        counts, cache) +
        validate_springs(&springs[1..], counts, cache)
    } else if springs.is_empty() {
        counts.is_empty() as usize
    } else if counts.is_empty() && springs.contains('#') ||
            springs.len() < counts[0] ||
            springs[..counts[0]].contains('.') {
        0
    } else if springs.len() == counts[0] {
        (counts.len() == 1) as usize
    } else if springs.chars().nth(counts[0]) == Some('#') {
        0
    } else {
         validate_springs(&springs[counts[0]+1..],
&counts[1..], cache)
    };
    cache.insert(cache_key, result);

    result
```

Sous-Vide Beefburgers

With jack-in-the boxes firmly at the front of my mind, what else was I going to serve but burgers? Hopefully these are a touch above the usual fare at their fast-food based inspirations.

Ingredient	Amount	
Minced Beef, freshly ground	500	g
Small onion, finely chopped	1	
Salt	20	g
Emmental Cheese, sliced	4	Slices
Bacon, streaky	8	Rashers
Brioche Rolls	4	
Gem Lettuce	1	Head

Method

1. Set the sous vide water bath to 60 degrees celsius
2. Mix beef, onions and salt thoroughly, before separating into four equal-sized pieces.
3. Fashion each piece into a ball then press gently into a hamburger shape. Seal each burger into its own sous vide pouch, then place in the water bath. Allow to cook for at least 45 minutes.
4. Separate and wash the lettuce leaves
5. Bring a cast iron frying pan to as high a temperature as possible. While it is heating, remove the burgers from their bath, and place on kitchen paper to drain.
6. Split and toast the brioche buns, and cook the bacon in the pan until crispy. Remove and drain on kitchen paper.
7. Sear the burgers for 45s on each side, enough to give a good browning, but no more.
8. Remove from heat and assemble the completed burger in the toasted buns.

DAY 13

Point of Incidence

Day 13
Thou dost beguile the world...

I really enjoy the problems where, when given a matrix, we have to compress it in some way to reach a conclusion. 2021's Transparent Origami was a highlight, and this reminds me of that one more than just a little bit. This time, the folding line may or may not be directly across a row or column, but it doesn't add a huge amount of complexity.

I'd already guessed part 2 after glancing at the input and seeing a grid that almost, but not quite, had a very obvious symmetrical line.

```
fn score_mirror(grid: &str, find_fn: fn(&Vec<String>) ->
            Option<usize> ) -> usize {
    let lines = grid.split('\n').map(
            |c| c.to_owned()).collect::<Vec<String>>();
    let vertical_reflection = find_fn(&lines).unwrap_or(0);
    let horizontal_reflection =
            find_fn(&reflect_grid(&lines)).unwrap_or(0);
    vertical_reflection + 100* horizontal_reflection
}

fn find_reflection_line(grid: &Vec<String>) -> Option<usize>
{
    let mut symmetries  = line_reflection_points(&grid[0]);
    for line in grid {
        let new_symmetries = line_reflection_points(line);
        symmetries = symmetries.intersection(
          &new_symmetries).copied().collect::<HashSet<_>>();
    }
    symmetries.iter().next().copied()
}

fn find_smudge_line(grid: &Vec<String>) -> Option<usize> {
    let mut symmetry_counts = HashMap::new();
    for new_symmetries in grid.iter().map(
    line_reflection_points) {
        for symmetry in new_symmetries {
            *symmetry_counts.entry(symmetry)
                 .or_insert(0) += 1;
        }
    }
    symmetry_counts.iter().find_map(
      |(k,v)| if *v == grid.len() -1 {Some(*k)} else {None})
}
```

Duck A L'Orange

From mirrors, we go to Shakespeare's Sonnet number 3, a poem instructing its subject to look in a mirror. The sonnet itself was recorded by someone who may or may not be Mark Oliver Everett and released as *Sonnet Number 3 (Like a Duck)* in the early noughties. And so we get to Duck a l'Orange.

Ingredient	Amount	
Whole Duck	1	
Large Orange	2	
Kosher Salt	2	tsp
Bay Leaf	2	
Onion	1.5	
Orange Liqueur	60	ml
Red Wine	75	ml
Orange Marmalade	3	tbsp
Cornflour	2	tsp

Method

1. Pre-heat the oven to 210C. Peel the zest from one orange with the vegetable peeler and trim half into into thin strips, discarding the rest

2. Place half the orange used above in the cavity of the duck, along with one onion, chopped into wedges and the bay leaves,, before pricking it all over with a fork.Place in a roasting tin, season with salt and pepper, and roast for 45 minutes.

3. Remove the duck from the oven, drain the fat from the roasting pan, and return for a further 45 minutes, before removing from the pan and leaving to stand covered loosely with foil

4. Slice the remaining onion thinly, then drain all but one tablespoon of fat from the roading pan, and place it over the hob, frying the onion for five minutes, striating frequently.

5. Add the orange liqueur and wine to the plan, bringing to a low simmer. Add 150ml of water and continue to simmer for 2 minutes, stirring to dislodge anything stuck to the pan

6. Pour the contents of the roasting tin through a sieve into a saucepan, Mix the cornflour with a small amount of cold water to make a slurry, then add it and the marmalade to the pan. Cook for a few minutes more, until the sauce is thickened. Add the reserved zest from stage 1.

7. Carve the duck at the table, pouring over the sauce once plated. Serve with roasted new potatoes and green beans.

DAY 14

Parabolic Reflector Dish

Day 14
Rotational Symmetry

What I really want to know is: Surely after a million rotations, the square rocks have been bashed into enough times to have their corners worn down and they're now round too?

This puzzle gave me flashbacks to last year's Pyroclastic flow; two puzzles whereby a bunch of rocks are being thrown around for way longer than is entirely reasonable. I'd enjoyed the game of not-quite-tetris, and I liked this one too.

```
#[derive(Debug,Clone,Copy, PartialEq, Eq)]
enum Rock {
    Round,
    Square
}
```

```
fn main() -> Result<(), Box<dyn Error>>{
    let data = get_daily_input(14,2023)?;
    let mut arena: HashMap<Coordinate<i64>, Rock> =
         parse_data(&data);
    let boundary = Coordinate{
        x: arena.keys().map(|c| c.x).max().unwrap(),
        y: arena.keys().map(|c| c.y).max().unwrap()
    };

    roll_rocks(&mut arena, &boundary, Direction::North);
    println!("Part 1: {}",
         support_beam_load(&arena, &boundary));

    let mut cycles = 0;
    let mut seen : HashMap<u64, i128> = HashMap::new();
    let mut state = hash_arena(&arena, &boundary);
    while !seen.contains_key(&state) && cycles < 200{
        seen.insert(state, cycles);
        cycle_rocks(&mut arena, &boundary);
        cycles +=1;
        state = hash_arena(&arena, &boundary);
    }
```

Borscht

It's hard to come up with a recipe based on tumbling rocks until a mirror ends up in the right shape. I've taken inspiration from last year's Pyroclastic Flow puzzle instead, with a dish inspired by the origins of Tetris.

Ingredient	Amount	
Minced Pork	450	G
Beetroot, medium, peeled and grated	3	
Carrots, peeled and grated	3	tsp
Baking potatoes	3	
Red Cabbage	0.5	head
Diced Tomatoes, fresh	100	g
Onion, chopped	1	
Garlic, minced	3	Cloves
Sour Cream	100ml	tsp
Tomato puree	150g	

Method

1. Over a medium heat, fry the minced pork until just starting to brown

2. Bring around 450ml water to boil, and add the mince and beetroot and simmer for 10 minutes.

3. Add the carrots and potatoes and cook until the potatoes start to soften, around a further 10 minutes. Add the cabbage and tomatoes.

4. Fry the onion until soft, then add the tomato puree and 150ml water, mixing thoroughly before adding the mixture to the saucepan.

5. Add the garlic, stir and leave to stand

6. Serve in bowls, topped with the sour cream, accompanied with sourdough.

DAY 15

Lens Library

Day 15
Making a real hash of things

I haven't thought about the fundamentals of how a hash map is implemented in a long time. Probably not since my first semester of university where an achingly cool lecturer with long, curly grey hair, a goatee and an unbearably bright tie-dye t-shirt attempted to knock some knowledge into me. It still amazes me how much I recall that I haven't touched in so many years.

This was a fun little exercise, though I didn't use a linked list to represent my data storage, that would have been a little too much like hard work.

```
fn part2(data: &str) -> usize {
    let mut hm: [Vec<Lens>; 256]  =
                                std::array::from_fn(|_| vec![]);
    for step in data.split(',') {
        if step.contains('-') {
            hm[checksum(&step[..step.len()-1])]
               .retain(|e| e.label != step[..step.len()-1]);
        }
        else if step.contains('=') {
            let label = &step[..step.chars()
                .position(|c| c == '=').unwrap()];
            let value = step[step.chars()
                .position(|c| c == '=').unwrap() +1 ..]
                .parse::<usize>().unwrap();
            let cs = checksum(label);
            let mut found = false;
            hm[cs].iter_mut().for_each(|l|
                  if l.label == label {l.value = value;
                  found = true} );
            if ! found {
                hm[cs].push(
                   Lens{label: label.to_string(), value})
            }
        }
    }
    hm.iter().enumerate().map(
        |(box_number, lens_list)|
            lens_list.iter().enumerate().map(
                    |(i, lens)|  (box_number+1) * (i +1) *
                                 lens.value).sum::<usize>()
    ).sum()
}
```

Corned-Beef Hash

What else? Though it turns out that Corned Beef means different things in different parts of the world. Don't blame me if this turns out otherwise than delicious, if your corned beef doesn't consist of sliced blocks of minced beef held together by a mystery substance.

Ingredient	Amount	
Corned Beef, diced	350	g
Potatoes, peeled and diced	500	g
Onion, chopped	1	
Worcestershire Sauce	2	tbsp
Sunflower Oil	2	tbsp

Method

1. Put the potatoes in a saucepan, cover with water, and bring to the boil, boiling for just long enough to cook the outer surfaces, 6-7 minutes, before draining.

2. Heat the oil in a cast-iron skillet over a medium heat, then add the onion and fry for 3-4 minutes.

3. Add the potatoes and corned beef, pressing into the pan to form a loose patty.

4. Cook for 5 minutes before turning, trying not to break up the mixture too much.

5. Drizzle over the Worcestershire sauce and cook for a further 5 minutes

6. Serve with baked beans.

DAY 16

The Floor Will Be Lava

Day 16
Working Up a Real Lava

What's that, Lassie? Another problem that reminds me of Mine Cart Madness? This time we're bouncing a beam of light around the place, and somehow, by the miracle of Elf magic, no matter how man times it gets split it doesn't diminish. I can't help feeling like there's some kind of quantum entanglement nonsense going on.

Loved the test data today, because it did something I was expecting it not to do: include a condition that would break the whole thing, in this case a loop. In previous years that would have been sneakily inserted into the real data ready to trap unsuspecting coders.

```
fn part1(arena: &HashMap<Coordinate<i32>, Tile>,
starting_point: Particle) -> usize {
    let mut unvisited: VecDeque<Particle> = VecDeque::new();
    let mut visited = HashSet::new();
    let mut seen_state = HashSet::new();

    let boundary: Coordinate<i32> = Coordinate{
        x: arena.keys().map(|c| c.x).max().unwrap(),
        y: arena.keys().map(|c| c.y).max().unwrap()
    };

    unvisited.push_back(starting_point);
    while ! unvisited.is_empty() {
        let current_loc = unvisited.pop_front().unwrap();
        if current_loc.location.x < 0 ||
current_loc.location.x > boundary.x ||
        current_loc.location.y < 0 || current_loc.location.y
> boundary.y ||
        seen_state.contains(&current_loc){
            continue;
        }
        visited.insert(current_loc.location);
        seen_state.insert(current_loc);
        match
arena.get(&current_loc.location).unwrap_or(&Tile::Empty) {
            Tile::Empty => {unvisited.push_back(Particle{
               location:current_loc.location.neighbour(
                    current_loc.heading),
               heading: current_loc.heading});},
            Tile::Mirror(mirror_direction) => {
                match mirror_direction {
                    TileDirection::NorthWestSouthEast => {
                        match current_loc.heading {
                            Direction::North =>
```

Chocolate Lava Cake

I'm told that cooking with volcanoes is a thing, However if you lack an actual volcano in your kitchen, this is probably the closest you're going to get.

Ingredient	Amount	
Dark Chocolate	100	g
Butter	100	g
Light soft brown sugar	150	g
Large Eggs	3	
Flour	50	g
Vanilla Essence	0.5	Tsp

Method

1. Heat the oven to 200C, and grease 6 small pudding moulds.
2. Melt the butter and chocolate together in a microwave, stir until smooth and then leave to stand for 15 minutes.
3. Whisk in the sugar, followed by the eggs, one at a time, followed by the vanilla essence and finally the flour
4. Divide the mixture between the moulds, and bake for 10-12 minutes
5. Serve immediately turning out onto plates , topped with single cream.

DAY 17

Clumsy Crucible

Day 17
Winding a Wobbly Way

The word crucible has an entirely different connotation to my mind than anyone who didn't grow up in 1980s England. Neither Arthur Miller, nor the small vessel used in chemistry spring to mind first of all, but the Northern English venue used as the host of the World Snooker Championships.

Not that that had any bearing whatsoever on today's task; instead it's a Djikstra-style node search, for which the nuance is identifying the node as being a combination of location, distance and steps remaining. A new twist on an old problem, one I enjoyed very much.

```
fn part1(grid: &HashMap<Coordinate<i32>,i32>, min_move:
i32, max_move:i32) -> i32 {

    let mut repeated_states = HashMap::new();
    let mut unconsidered = BinaryHeap::new();

    let goal = Coordinate{
        x: grid.keys().map(|c| c.x).max().unwrap(),
        y: grid.keys().map(|c| c.y).max().unwrap()
    };

    unconsidered.push(ScoredItem{cost:0 , item:
Crucible{facing: Direction::East, location: Coordinate {
x:0, y:0}, steps_taken: 0}});
    unconsidered.push(ScoredItem{cost:0 , item:
Crucible{facing: Direction::South, location: Coordinate
{ x:0, y:0}, steps_taken: 0}});


    while let Some(current_square)  = unconsidered.pop()
{

        let best_for_square =
repeated_states.get(&current_square.item).unwrap_or(&i32
::MAX);
        if *best_for_square <= current_square.cost  {
            continue;
        }
        repeated_states.insert(current_square.item,
```

Trifle

What better match for a wobbly item winding its way through the streets of Elfville than this traditional English dish made with jelly, custard and cream?

Ingredient	Amount	
Sponge fingers,	6	
Packet Hartley's Strawberry Jam	1	
Bird's Custard Powder	3	tbsp
Milk	500	ml
Sugar	2	tbsp
Whipping Cream	300	Ml
Sherry	2	tbsp

Method

1. Layer the sponge fingers across the bottom of a large glass bowl, and pour over the sherry.

2. Make the jelly up according to the instructions on the packet, and pour over the sponge fingers. Refrigerate until set, at least 1 hour.,

3. Add the custard powder to a small amount of the milk and mix to form a paste. Add to a saucepan with the remaining milk and the sugar,, and bring to a low simmer, stirring constantly until the custard reaches a thick consistency. Pour over the jelly, and return to the fridge to cool for a further hour,

4. Whip the cream until stiff, and layer over the custard.

5. Serve.

DAY 18

Lavaduct Lagoon

Day 18
Lost in a lagoon of ludcrous proportion

I can't help wondering if the elves have mixed up metres for centimetres. Fortunately we're on a magical island of unmeasurable proportions, else we'd be stuck digging out pretty much the whole landmass of the planet to hold the lava needed.

Shoelace theorem, which I wasn't familiar with before day 10, came in extremely handy today.

```
fn answer(instructions: &Vec<Instruction>) -> i128 {

    let mut position = Coordinate{x:0, y:0};
    let mut locations = Vec::new();
    let mut perimeter = 0;
    for instruction in instructions {
        locations.push(position);
        perimeter += instruction.distance;
        let delta = match instruction.direction {
            Direction::North => {Coordinate{x:0, y:-
instruction.distance}},
            Direction::East =>
{Coordinate{x:instruction.distance, y:0}},
            Direction::South => {Coordinate{x:0,
y:instruction.distance}},
            Direction::West => Coordinate{x:-
instruction.distance, y:0},
            _ => unimplemented!()};
        position += delta;
    }
    // Shoelace Theorem - gives area including perimeter
    // However this is an overestimate, as we want the
area bounded by
    // the centre of each 1x1 block, not the whole block
    let mut area = 0;
    for i in 0..locations.len()-1 {
        area += locations[i].x * locations[i+1].y
            - locations[i].y * locations[i+1].x;
```

Blue Lagoon

There was one obvious pairing with today's task - a Blue Lagoon cocktail.

Ingredient	Amount	
Blue Curaçao	100	ml
Vodka	100	Ml
Lemon, juiced	1	
Lime, juiced	1	
Orange, juiced	1	
Maraschino cherries	8	
Syrup	1	tsp
Soda Water	200	ml
Ice cubes	Handful	
Crushed ice		

Method

1. Place a number of ice cubes in a large jug, adding the remaining liquid ingredients except soda water, stirring vigorously
2. Fill 4 hi-ball glasses with crushed ice, and pour over the cocktail.
3. Top up the glasses with soda water
4. Garnish with the cherries.
5. Server immediately.

DAY 19

Aplenty

Day 19
Happy XMAS (sorting is over)

Build a machine to classify parts according to four different dimensions. Then extend it to classify all possible parts. Easy enough, right?

Today's solution had a large number of bugs in it that it turned out made absolutely no difference to the solution, due to the way in which I was throwing away map lookups that had no match. I might call that a win; this is Advent of Code. The code answer that gets the right result wins, regardless of whether it works for more general cases.

```
fn main() -> Result<(), Box<dyn Error>>{
    let data = get_daily_input(19,2023)?;
    let mut fields = data.split("\n\n");
    let ruleset = build_ruleset(fields.next().unwrap());
    let parts = fields.next().unwrap();
    println!("Part 1: {}", part1(&ruleset, parts));
    println!("Part 2: {}", part2(&ruleset));
    Ok(())
}

fn part1(ruleset: &HashMap<String,Vec<ComparisonRule>>,
items: &str) -> i128 {
    items.lines().map(parse_machine_part).filter_map(
            |i| process_item(&i, ruleset)).sum::<i128>()
}

fn build_ruleset(data: &str) ->
HashMap<String,Vec<ComparisonRule>> {
    let mut rules = HashMap::new();
    for line in data.lines() {
        let splitter = line.find('{').unwrap();
        let label = line[..splitter].to_string();

rules.insert(label,line[splitter+1..line.len()-1].split(
',').map(parse_rule).collect::<Vec<_>>() );
    }
    rules
}
```

Egg-Fried Rice

We have 256 trillion parts to sort. I can't fit that many of anything at a plate at a time; the closest I could manage are a few thousand grains of rice. I hope that will suffice. This is a non-traditional version of the dish;

Ingredient	Amount	
Long-grain rice	250	g
Onion, finely chopped	1	
Eggs, beaten	4	
Peas, frozen	50	g
Sweetcorn, frozen	50	g
Bacon lardons	100	G
Soy Sauce	Splash	

Method

1. Cook the rice according to the instructions on the packet, then spread out to dry
2. Heat 2tbsp of oil in a frying pan, and add the bacon and onion, frying until they are soft.
3. Add the rice, stirring to toast for 3 minutes and move to the side of the pan
4. Add a further 1 tbsp of oil to the pan, and tip in the eggs, stirring rapidly to scramble a little before stirring in the rice.
5. Stir in the peas, sweetcorn and soy sauce and allow to heat through.
6. Serve immediately

DAY 20

Pulse Propagation

Day 20
A Calculator by any other name

As a teenager, I had an already-ancient book that described how to construct a basic calculator using nothing but transistors consisting flip-flop circuits. This reminded me very closely of that.

Part 2 was easier than some reverse engineering tasks had been, merely looking at the inputs of the penultimate node and working out when they would all match.

```
impl CommunicationModule {
    fn receive(&mut self, pulse: &Pulse, sender: &str)
              -> Pulse{
        match self.kind {
            ModuleType::FlipFlop => {
                if *pulse == Pulse::Low &&
                  self.state == ModuleState::Off {
                    self.state = ModuleState::On;
                    Pulse::High
                } else if *pulse == Pulse::Low &&
                  self.state == ModuleState::On {
                    self.state = ModuleState::Off;
                    Pulse::Low
                } else {
                    Pulse::None
                }
            },
            ModuleType::Conjunction => {
                self.memory
                  .insert(sender.to_string(), *pulse);
                if self.memory.values().all(
                        |x| *x == Pulse::High) {
                    Pulse::Low
                } else {
                    Pulse::High
                }
            }
            ModuleType::Broadcast => *pulse,
        }
```

Habas Fritas

Given the title, this had to be a recipe involving pulses of some kind; the trouble is that I'm not a big fan of most pulses. These, however, make a decent snack.

Ingredient	Amount	
Dried Split Broad Beans	125	g
Olive oil	4	Tsp
Salt Flakes	1	tsp

Method

1. Soak the broad beans for 6-8 hours or overnight.
2. Preheat the oven to 180C.
3. Drain the broad beans, rinse well, place in a saucepan and cover with about 500ml of cold water.
4. Bring the broad beans to a boil over a medium-high heat and boil for about 7 minutes. This is just to soften them somewhat (so you should be able to pierce them with a fork, but they should not be remotely mushy). Without this step, I find the roasted broad beans to be just a bit too hard.
5. Drain the broad beans and allow to cool a little.
6. Mix the salt and olive oil and then toss the broad beans in this mixture.
7. Spread the broad beans out on the baking tray and place in the oven to roast for about 25-30 minutes or until golden.