Day 09

Snowballs, Traits, and Furious Santa

The Story

The elves trudged into the North Pole office, their little faces as frosty as leftover cocoa. The holiday crunch was bad enough, but Blitzen's latest antics had turned it into a nightmare. As the self-appointed "Tech Lead," Blitzen wouldn't stop yapping about optimizing Rust code that didn't need optimizing.

The Story

Meanwhile, the real issues in Santa's GitHub repo—the gift filtering bug and the broken nice-kid calculator—were piling up like unwrapped presents on December 24th.

The Story

The elves slumped into their desks, morale at absolute zero. That's when Santa stomped into the room, red-faced and gripping his enormous candy cane staff. His expression said it all: Unhinged Santa Mode Activated.

The Story

"WHAT IN THE NAME OF PRANCER'S LEFT HOOF IS GOING ON HERE?!" Santa bellowed. "Blitzen, what"s this I hear about you trying to write your own grep?!"

Blitzen froze, mid-boast about his hand-rolled, regex-powered masterpiece. "Uh, well, I thought—"

The Story

"You thought? You thought I needed a worse grep when the sleigh's landing system is malfunctioning?!" Santa's eyes blazed like LEDs on overclock. "Do you know where we landed yesterday?! Florida. FLORIDA, BLITZEN! You ever tried explaining a snow-laden sleigh to a guy in flip-flops?!"

Santa turned to the rest of the elves. "Everyone, listen up. Blitzen... you’re FIRED!"

The Story

The room gasped. Even the code compiler on the corner workstation seemed to pause in shock. Blitzen’s antlers drooped.

"Well, not fired," Santa clarified, stroking his beard. "But you're off tech lead. Effective immediately, you’re on..." he smirked, "Candy Cane API maintenance. Enjoy your new repo."

The Story

The elves stifled giggles as Blitzen sulked away. Santa clapped his hands, bringing their attention back to the matter at hand.

The Story

"Now, here's the real problem, folks. The sleigh's landing system needs snow data, but look at what I got instead." He waved a crinkled printout. "Snow weights! In kilograms and pounds! I don't know what any of this means! What I need is actionable snow data—converted into snowballs."

The elves exchanged worried glances.

The Story

"I want you to implement the `From<T>` trait to convert these." Santa jabbed at the paper. "We've got `SnowKg(pub f32)` and `SnowLb(pub f32)`, but I need them in a usable format: `Snowball(pub i64)`! And make it snappy! Christmas Eve doesn't debug itself!"

The Story

The room buzzed with renewed energy. The elves, though still peeved at Blitzen, couldn't help but feel the spark of a good challenge. Snowball-ready data? Time to roll up their sleeves—and their structs.

Santa grinned as he marched out. "And if anyone so much as mentions rewriting ls in Rust, you're on tinsel-wrapping duty."

Your Mission

- Implement the `From<T>` trait for the `SnowKg` and `SnowLb` types to convert them into `Snowball` type.
- The `Snowball` type should contain the number of snowballs that can be made from the given weight.

Your Mission

- The results should be rounded down or up to the nearest whole number.
- Make sure you use the `SNOWBALL_WEIGHT_KG` and `SNOWBALL_WEIGHT_LB` constants to calculate the number of snowballs.

Hints

If you're stuck or need a starting point, here are some hints to help you along the way!

- Use the `SNOWBALL_WEIGHT_KG` and `SNOWBALL_WEIGHT_LB` constants to find the number of snowballs from a weight.
- Implement the `From<T>` trait for each type. e.g. `impl From<WeightInKg> for Snowball`.

Hints

- Inside the `impl From<T> for Snowball` block, calculate the number of snowballs from the input type. e.g.

```
fn from(weight: SnowKg) -> Self {
    let value = "calculate the number of snowballs from weight";
    SnowBall(value)
}
```

Hints

- Use `round()` to round the result to the nearest whole number. e.g. `value.round()`.
- Convert the result to an integer using `as i64`.