--- Day 7: Recursive Circus ---

Wandering further through the circuits of the computer, you come upon a tower of programs that have gotten themselves into a bit of trouble. A recursive algorithm has gotten out of hand, and now they're balanced precariously in a large tower.

One program at the bottom supports the entire tower. It's holding a large disc, and on the disc are balanced several more sub-towers. At the bottom of these sub-towers, standing on the bottom disc, are other programs, each holding their own disc, and so on. At the very tops of these sub-sub-sub-...-towers, many programs stand simply keeping the disc below them balanced but with no disc of their own.

You offer to help, but first you need to understand the structure of these towers. You ask each program to yell out their name, their weight, and (if they're holding a disc) the names of the programs immediately above them balancing on that disc. You write this information down (your puzzle input). Unfortunately, in their panic, they don't do this in an orderly fashion; by the time you're done, you're not sure which program gave which information.

For example, if your list is the following:

```
pbga (66)
xhth (57)
ebii (61)
havc (66)
ktlj (57)
fwft (72) -> ktlj, cntj, xhth
qoyq (66)
padx (45) -> pbga, havc, qoyq
tknk (41) -> ugml, padx, fwft
jptl (61)
ugml (68) -> gyxo, ebii, jptl
gyxo (61)
cntj (57)
```

...then you would be able to recreate the structure of the towers that looks like this:

```
                gyxo
              /
         ugml - ebii
       /      \
      |         jptl
      |
      |         pbga
     /        /
tknk --- padx - havc
     \        \
      |         qoyq
      |
      |         ktlj
       \      /
         fwft - cntj
              \
                xhth
```

In this example, `tknk` is at the bottom of the tower (the bottom program), and is holding up `ugml`, `padx`, and `fwft`. Those programs are, in turn, holding up other programs; in this example, none of those programs are holding up any other programs, and are all the tops of their own towers. (The actual tower balancing in front of you is much larger.)

Before you're ready to help them, you need to make sure your information is correct. What is the name of the bottom program?

Your puzzle answer was `uownj`.

--- Part Two ---

The programs explain the situation: they can't get down. Rather, they could get down, if they weren't expending all of their energy trying to keep the tower balanced. Apparently, one program has the wrong weight, and until it's fixed, they're stuck here.

For any program holding a disc, each program standing on that disc forms a sub-tower. Each of those sub-towers are supposed to be the same weight, or the disc itself isn't balanced. The weight of a tower is the sum of the weights of the programs in that tower.

In the example above, this means that for `ugml`'s disc to be balanced, `gyxo`, `ebii`, and `jptl` must all have the same weight, and they do: `61`.

However, for `tknk` to be balanced, each of the programs standing on its disc and all programs above it must each match. This means that the following sums must all be the same:

- `ugml` + (`gyxo` + `ebii` + `jptl`) = 68 + (61 + 61 + 61) = 251
- `padx` + (`pbga` + `havc` + `qoyq`) = 45 + (66 + 66 + 66) = 243
- `fwft` + (`ktlj` + `cntj` + `xhth`) = 72 + (57 + 57 + 57) = 243

As you can see, `tknk`'s disc is unbalanced: `ugml`'s stack is heavier than the other two. Even though the nodes above `ugml` are balanced, `ugml` itself is too heavy: it needs to be `8` units lighter for its stack to weigh `243` and keep the towers balanced. If this change were made, its weight would be `60`.

Given that exactly one program is the wrong weight, what would its weight need to be to balance the entire tower?

Your puzzle answer was `596`.

Both parts of this puzzle are complete! They provide two gold stars: **

At this point, you should return to your advent calendar and try another puzzle.

If you still want to see it, you can get your puzzle input.

You can also [Share] this puzzle.