Code Jam 2012 - Qualification Round

Analysis: Dancing With the Googlers

Clever contestants will have noticed that this problem might be based on an actual television show. The author of the problem found himself watching an episode of *Dancing With the Stars* (the show and its distributor, BBC Worldwide, do not endorse and are not involved with Google Code Jam), and having forgotten whether any of the contestants had earned a score of at least 8 from any of the judges.

The judges on that show very often agree within 1 point of each other, and it really is surprising when they disagree by so much. There are a few key observations that help us solve this problem:

1. Any non-surprising score is expressed uniquely by an unordered triplet of judges' scores. 21 must be 7 7 7; 22 must be 7 7 8; 23 must be 7 8 8.
2. There only a few kinds of surprising scores. 21 could be 6 7 8; 22 could be 6 8 8; 23 could be 9 7 7. There's a repeating pattern of surprising scores that extends from 2 (0, 0, 2) to 28 (8, 10, 10).
3. The repeating pattern stops at 2 and 28 because 1 and 29 can't be surprising: we can't use (-1, 1, 1) for 1: -1 is not allowed as a score. We also can't use (9, 9, 11) for 29.

Putting all of these facts together, we can easily construct a mapping from each total score to its *best result* if it's surprising, and to its best result if it isn't surprising:

```
unsurprising(0) = 0
unsurprising(1) = 1
unsurprising(2) = 1
unsurprising(n) = unsuprising(n-3) + 1
unsurprising(n) = ceiling(n/3), for 0 <= n <= 30

surprising(2) = 2
surprising(3) = 2
surprising(4) = 2
surprising(5) = 3
surprising(n) = surprising(n-3) + 1
surprising(n) = ceiling((n-1)/3) + 1, for 2 <= n <= 28
```

Many contestants wrote an initial solution that didn't take into account that 0, 1, 29 and 30 can't be surprising, and so failed one of our sample inputs. That's what they're there for!

One way to build the tables of unsurprising(n) and surprising(n) that doesn't require so much thought could involve writing three loops to go through all possible sets of judges' scores, checking whether each combination of three was valid or surprising, and building the maps that way.

Now, for each value t_i we have one of three cases:

- unsurprising(t_i) >= p. This Googler can be "good" (i.e. have a maximum score of at least p) even with an unsurprising triplet.
- 2 <= t_i <= 28 and surprising(t_i) >= p > unsurprising(t_i). This Googler can be "good" only by using a surprising triplet.
- Otherwise, the Googler can't be "good".

A simple greedy algorithm works: take all of the Googlers of the first type, and as many as possible (at most S) of the second type.