COP 4516 Spring 2025 Week 2 Individual: Greedy (Solution Sketches)

Balloons
Sort by DIFFERENCE in distance of the balloons from A and B, with larger distance going first. So say team1 is 10 away from A and 50 away from B, this comes before team2 which is 1000 away from A and 990 away from B. Basically, since all teams get balloons, if team1 gets a balloon from B, that's 40 more than optimal while for team2, getting a balloon from A is only 10 more than optimal. So, sort in this order, then give priority to the teams in this order, getting them all balloons from the shorter location. If at any point balloons from one location run out, hand out all other balloons for the remaining problems from the place that has balloons.

Conference
The payment is equal to a binary number with some ones bits turned on. A property of binary numbers is that turning on the most significant bit is more value than turning on all the bits below it. Specifically, the sum of the terms $2^0 + 2^1 + 2^2 + 2^k = 2^{k+1} – 1$. Thus, it’s always better to schedule a conference that starts earlier, even if it’s just for one day. Since the conferences all start on unique days, this creates a unique sorting order. Sort the conferences by start date, and then go through the sorted list, scheduling a conference if the venue is free at that time. The time a venue becomes free after booking a conference on day d that lasts s days is day d + s. No conference is too long, so we can add the value of each conference by adding each day one at a time…

Fruit
For each i, calculate ceiling (sum of fruits sold first i days)/i. At the very least, you need that many fruits each day to make it through day i with fruit. Of all of these values over all possible i, take the maximum. Then, rerun the simulation getting this many fruits each morning and store the maximum ever leftover...Example: 3, 8, 2, 5, 4, 6. Fruits needed for day 1: 3, for days 1,2: 6, days 1,2,3:5, days1-4: 5, days1-5: 5, days1-6: 5. Max is 6. Now simulate: day 1 left with 3 fruits (6-3), day 2 left with 1 fruit (3+6-8), day 3 left with 5 fruits(1+6-2), day 4 left with 6 fruits (5+6-5), day 5 left with 8 fruits (6+6-4), day 6 left with 8 fruits (8+6-6), So most you ever store is 8.

Polling
Keep an ordered map from String->Integer of name to # of votes. As each vote is read in, also keep track of the maximum # of votes any person receives. (Just keep one extra max variable and update it, if necessary.) After reading in the data, the # of maximum # of votes any person received is known. Thus, go through the ordered map in the natural order and just print a name if that person got the maximum # of votes. If there is no ordered map (Python), just use a regular map, and then instead of printing, when you go through the map, just copy each key that has the maximum number of votes into a list, sort the list and output. Lots of different ways to do this one. Key is to do O(lg n) or better individual operations.