Problem 4:

Pre-Implementation Notes:

With this problem it can simply be understood as the minimum number of times you need to re-complete the most optimal quests in the given time frame, as that is exactly what the value of `k` constitutes. It puts a limitation on the number of times that I can recomplete the most rewarding quest.

There are three cases:

Case: `k = Impossible`

This case is when, even if i redo the most rewarding quest every single day, i won't be able to finish the quest

Case: `k = Infinity` This case is, if i complete each quest once and that get's me above the coins I need, then the value of `k` is unimportant to whether i can hit my coin goals or not.

Case: `k` is an integer

This case is there exists the maximum value `k` that I can have in order to still hit my goals.

With this in mind, my initial solution is to check for the first 2 cases, and for the third case i'm going to binary insert the value of `k` until an optimal solution is found.

Possible `k` values are `0 <= k <= n - 2`

To think about the largest value of `k` that would be the worst case scenario where you NEED to complete the most rewarding quest twice. Once at hte start and once at the end. That would give you the value k to be `k = n - 2`.

Post-Implementation Notes:

- During implementation, i realised that you could just calculate how many quests you want to do `k` spaces apart just by diving how many spaces are left.
- Binary insert gave me a lot of bugs, especially with indexing, i should practice this more
- Other then that, my original solution worked flawlessly within the desired time