Problem:
You live in a remote settlement where bread sellers come through periodically at irregular intervals. Whenever you buy fresh bread, it lasts for 30 days until it becomes too stale to eat. Your family eats one loaf of bread per day.

You are given a calendar of when the bread sellers will be visiting over the coming days and the price each bread seller will charge per loaf. You currently have 10 fresh loaves, and at the end of the calendar you'll get a bunch of free bread, so you won't need to have any left on hand. Write a function that tells you how much bread to buy from each of the sellers

Input:
`total_days`, an integer, the number of days in the calendar until the free bread arrives
`sellers`, a list of pairs of integers (`day, price`). Each pair represents one bread seller
    The `day` is how many days from the start until the seller arrives
    The `price` is the price to buy each loaf of bread from this seller, in pennies

Output:
`purchases`, a list of integers of the same length as `sellers`. Each integer is how many loaves you should buy from each seller
Or `None`, if there is no solution that does not force your family to eat stale bread at some point

You should output the purchase plan that minimizes the total cost. In case of ties, output the plan that requires buying from the fewest number of different sellers, and within those choose the plan that buys more bread earlier

Requirements:
Please write a well formatted, easily readable python function `calculate_purchasing_plan(total_days, sellers)` in a file called `your_name.py` that solves this problem using only the standard python (i.e. no packages required).

Please include a comment at the top of the file with your name and email as well as a short paragraph describing why the algorithm you've provided should work.

Example:
```
>>> calculate_purchasing_plan(60, [(10,200), (15,100), (35,500), (50,30)])
[5, 30, 5, 10]
```