The Sushi Train Problem

At any given time, there are a varying number of chef's preparing sushi and adding trays to the sushi train. New trays are added to the train via a delivery belt from the kitchen, the chefs place new trays on the delivery belt which then transport the trays to the train. A smart delivery mechanism makes sure the new trays are added in the appropriate spot on the train.

The chefs prepare the trays in the kitchen and do not have visibility of the sushi train in the dining area. There is an overhead display which attempts to keep track of the number of trays on the train at any given time.

When a tray is placed on a train, it is 'scanned in' and when a tray is taken off the belt, it is 'scanned out'. **The number of trays on the display is simply the accumulated number of trays that have been scanned in minus the number of trays scanned out.**

It is known that the scanning mechanism is not 100% accurate and can sometimes missscan. As the trading day progresses, this results in a cumulative error and the display becomes less and less accurate.

The restaurant is open 24 hours a day, however at midnight any trays left on the belt are thrown out and the chefs start with a fresh set of ingredients.

It is also known that on average a tray stays on the belt for 90 minutes.
A tray will rarely remain on the belt for more than 3 hours; this time can be referred to as the "shelf life".

From 4pm onwards, the ingredients are not as fresh as they were at the start of the day, so the trays are sold at a discounted rate and the shelf life drops to 1.5 hours.

The following logic is used to correct for missed scans. Adjustment 1 is applied before adjustment 2.

Error Adjustment 1
At any point in time, if the cumulative number of trays scanned "*out"* does not equal at least the cumulative number of trays scanned "*in"* 3 hours prior (shelf life), we assume that one or more trays were removed but failed to scan, the difference is artificially added to the "*out*" scan at the current time point.
From 4pm onwards, a 1.5 hour shelf life is used instead of 3 hours.
Note: if it hasn't been 3 hours since the start of the day this adjustment is not applied.

Error Adjustment 2
At any given time if the count number of trays on the train becomes negative, we assume that one or more trays added to the train failed to scan in, so the difference is artificially added to the "*in*" scan 1.5 hours earlier.

Task

Write a script that accepts a file path as a command line argument.. The given file will contain the scans per minute from the start of the day up till the last time. **Calculate the output expected on the display screen at the final timestamp and write to stdOut.**

The output is a single integer representing the total number of trays on the sushi train *(accumulated number of trays that have been scanned in minus the number of trays scanned out)* at the last time after applying the adjustment logic.

You should only need to use standard C libraries.

Submission

A single archive file containing the:

- Source files
- Any required build configuration files
- Any required instructions to compile and run the code.

Data Input Structure

DATE, TIME, # SCANS IN, # SCANS OUT

Comma separated line for each timestamp e.g:

```
2022-06-28,05:45:00,1,0
2022-06-28,05:46:00,0,0
2022-06-28,05:47:00,0,0
2022-06-28,05:48:00,0,0
2022-06-28,05:49:00,0,0
2022-06-28,05:50:00,0,0
2022-06-28,05:51:00,0,0
2022-06-28,05:52:00,0,0
2022-06-28,05:53:00,2,0
2022-06-28,05:54:00,0,0
2022-06-28,05:55:00,0,0
2022-06-28,05:56:00,0,0
2022-06-28,05:57:00,0,1
2022-06-28,05:58:00,0,0
2022-06-28,05:59:00,0,0
```

Data Output Structure

```
2
```