Introduction

Build an application that would process infinite stream of events. It should perform two actions on every event received:

- Validate the checksum of only the payload component
- Calculate statistics of length of payload of events received within the last minute

Time and space complexity

The application would be run on a processor that could store in memory only the number of events generated in one minute.
Time complexity of the complete solution that calculates metrics for all captured events should be not worse than *O(nlogn)*, where *n* is number of processed events.
It is expected from a candidate to carefully choose data structures to meet these requirements and use built-in ones if available in a programming language of choice.

Input

The stream of events could be read from CSV file. You shouldn't assume that the events' timestamps are ordered – that is, an event with a later timestamp can appear before an event with an earlier timestamp in the stream, but you also have the freedom to reject or disregard the events that appear "too late" in the stream. Be prepared for duplicates, we can consider an event a duplicate of another if they both have the same id.
The samples of input could be found in this document:

Event structure

The table below presents the schema of the event.

Field name	Position in the event	Type	Example	Comment
Identifier	1	long integer	234	
Timestamp	2	long integer	1669734409	Unix Timestamp, seconds from January 1st, 1970
Payload	3	string	Content_1	Using ascii encoding
Checksum	4	Short integer	5	

Validate the checksum of the content

Calculate a check sum for the data stored in field payload of message, the algorithm is very simple, you take ascii code for every character of the event's payload, calculate the sum of them and calculate remainder of the sum divided by 10. In case of not successful validation your application should print "Event <event_id>: Invalid checksum".

Example no 1

For following list of events given as an input for the application

Identifier	Event ts (seconds)	Payload	Checksum
4	456	Box	7
5	466	AAA	1
6	506	xyz	3

The output would be

```
Event 4: 3, max 3, window ends 456
Event 5: Invalid checksum
… rest of the output follows ...
```

Calculate statistics of length of payload of events received within the last minute

Calculate a moving average and moving maximum of length of Payload of events received within the last minute. A window has start time and end time. The end time is based on the clock driven by the timestamps of received events. Calculated statistics should be printed out after event is processed, including event id and window end time. Event would be rejected if is received before the start time of current window. Every calculation performed should result in publishing a message to output using following template “Event <event_id>: average <calculated_average>, max <calculated_max>, window ends <timestamp_of_higer_end_of_window>”

Example no 2

Input

Identifier	Event ts (seconds)	Payload	Checksum
10	456	abcd	4
11	466	abcde	5
12	506	abcdef	7
13	520	a	7
14	570	abcde	5

Output

```
Event 10: average 4, max 4, window ends 456
```


```
Event 11: average 4.5, max 5, window ends 466
Event 12: average 5, max 6, window ends 506
Event 13: average 4, max 6, window ends 520
Event 14: average 3, max 5, window ends 570
```

Example no 3

Input			
Identifier	Event ts (seconds)	Content	Checksum
21	450	abcd	4
22	460	abcde	5
23	530	abcdef	7
24	460	a	7
25	570	abcd	4

```
Event 21: average 4, max 4, window ends 450
Event 22: average 4.5, max 5, window ends 460
Event 23: average 6, max 6, window ends 530
Event 24: Ignored, too late received, window starts 470
Event 25: average 5, max 6, window ends 570
```

Example no 4

Identifier	Event ts (seconds)	Content	Checksum
31	10	abcd	4
32	50	abcde	5
33	30	ab	5
34	20	abcdef	7
35	90	ab	5
36	120	a	7

```
Event 31: average 4.00, max 4, window ends 10
Event 32: average 4.50, max 5, window ends 50
Event 33: average 3.67, max 5, window ends 50
Event 34: average 4.25, max 6, window ends 50
Event 35: average 3.00, max 5, window ends 90
Event 36: average 1.50, max 2, window ends 120
```