Briefing

Please find below a coding problem we'd like you to solve. There is no “right” or “wrong” answer; rather we want to see how you code. So, make your code talk!

You should submit your solution in a way that makes it easy for us to use. We'd expect to see an Ant or Maven or Gradle build script. Unless you include documentation to tell us otherwise, we will expect to be able to build and run the solution with no changes.

It should be quick and easy for us to verify that the input does indeed lead to the expected output. Please use English as the primary language in your solution to enable us to verify the behaviour.

What are we looking for?

We want to see you can create production-quality code. This means naming conventions, coding style, sensible design and meaningful commenting. If you feel you can give your work to a brand new colleague with a minimal of hand-over, you've probably got it right.

We like TDD, so including unit tests is a good idea.

We realize that writing even a trivial application, and having it "production ready" is a lot of work - so it's okay to leave "ToDo" comments in your code. Show us the intent, but don't write too much boilerplate.

Show that you're coding something that needs to be maintainable.

Let us know how long you spent on it and where you did it.

What are we not looking for?

We don't expect you to use every design pattern you've ever heard of - only apply patterns when it makes sense to do so.

We don't expect you to build a user interface - a command line application or unit test is fine.

We're not expecting you to have optimized the solution for performance or memory size - readability is more important than performance.

The whole thing built in the shortest amount of time - it's not a race. We'd like to see what you'd do under normal conditions - if you run out of time, just say so.

Problem

We have an existing system for employees to submit booking requests for meetings in a meeting. We want you to implement a new system for processing batches of booking requests.

Input
Your processing system must process input as text.

- The first line of the input text represents the company office hours, in 24 hour clock format
- The remainder of the input represents individual booking requests. Each booking request is in the following format.
  [request submission time, in the format YYYY-MM-DD HH:MM:SS] [ARCH:employee id]
  [meeting start time, in the format YYYY-MM-DD HH:MM] [ARCH:meeting duration in hours]

A sample text input follows, to be used in your solution.

Input

```
0900 1730
2015-08-17 10:17:06 EMP001
2015-08-21 09:00 2
2015-08-16 12:34:56 EMP002
2015-08-21 09:00 2
2015-08-16 09:28:23 EMP003
2015-08-22 14:00 2
2015-08-17 11:23:45 EMP004
2015-08-22 16:00 1
2015-08-15 17:29:12 EMP005
2015-08-21 16:00 3
```

Output
Your system must provide a successful booking calendar as output, with bookings being grouped chronologically by day. For the sample input displayed above, your system must provide the following output.

Output

```
2015-08-21
```

```
09:00 11:00 EMP002

2015-08-22

14:00 16:00 EMP003

16:00 17:00 EMP004
```

Constraints

- No part of a meeting may fall outside office hours.
- Meetings may not overlap.
- The booking submission system only allows one submission at a time, so submission times are guaranteed to be unique.
- Bookings must be processed in the chronological order in which they were submitted.
- The ordering of booking submissions in the supplied input is not guaranteed.

Notes

- The current requirements make no provision for alerting users of failed bookings; it is up to the user to confirm that their booking was successful.
- Although the system that you produce may open and parse a text file for input, this is not part of the requirements. As long as the input text is in the correct format, the method of input is up to the developer.