Exercise: Clock

Basic functions

In the provided Elm project, you will find a module `Clock`, where you need to implement a function that operates on a representation of clocks (or a representation of time). We call a "clock" a tuple of type `(Int, Int)` which contains a time of day in the 24-hour format, that is, a value `(h, m)` where `h` is an integer between 0 and 23, both extremes included, and `m` is an integer between 0 and 59, again both extremes included.

The function you need to implement is `move`, which has the following signature:

```
move: Int -> (Int, Int) -> (Int, Int)
```

This function receives an arbitrary integer `delta` and a clock and must return the resulting clock of moving `delta` minutes.

For example, `move 30 (8, 45)` results in `(9, 15)`, while `move 30 (23, 45)` results in `(0, 15)`. As mentioned above, the `delta` parameter is an arbitrary integer and it can thus be negative or arbitrarily large. Therefore, `move -30 (0, 15)` should result in `(23, 45)`, while `move 2885 (8, 45)`, noting that a full day has 1440 minutes (24 times 60), should result in `(8, 50)`.

The provided Elm project contains also a module `Tests` with some unit tests.

End of exercise