Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
https://courses.softlab.ntua.gr/pl2/

Γλώσσες Προγραμματισμού II

Αν δεν αναφέρεται διαφορετικά, οι ασκήσεις πρέπει να παραδίδονται στους διδάσκοντες σε ηλεκτρονική μορφή μέσω του συνεργατικού συστήματος ηλεκτρονικής μάθησης `moodle.softlab.ntua.gr`. Η προθεσμία παράδοσης θα τηρείται αυστηρά. Έχετε δικαίωμα να καθυστερήσετε το πολύ μία άσκηση.

Άσκηση 3 Erlang για ραλλίστες

Προθεσμία παράδοσης: 10/1/2021

Ο σκοπός της συγκεκριμένης άσκησης είναι να σας εξοικειώσει με τη γλώσσα προγραμματισμού Erlang. Η παρακάτω άσκηση προέρχεται από το MIUP'2006, και είναι διαθέσιμη online εδώ. Την ακαδημαϊκή χρονιά 2011-12 δόθηκε ως άσκηση στο μάθημα των Γλωσσών Προγραμματισμού I — δείτε εδώ. Αυτό που σας ζητείται να κάνετε είναι να τη λύσετε σε Erlang. Ακολουθεί η εκφώνησή της στα αγγλικά.

You are about to participate as a co-driver in the next edition of the "Rallye de Monte-Carlo" (organized by ACM — Automobile Clube de Monaco). Before the actual race all drivers are allowed to run on the tracks of the course. In these reconnaissance drives, the co-drivers, who sit next to the drivers, write down shorthand notes on how to best drive the stage.

Based on your observations on these notes, you were able to create a map with the advisable speed limits of track sections. Passing by these locations over the speed limit is not recommended, since it can make your car crash. To assist your pilot, you need to devise a winning strategy based on the speed limits. And a nice computer program could be handy for this task...

Given a car rally track marked with speed limits at specific locations, your goal is to devise a strategy to reduce and increase the car speed such that you will run the track at the fastest possible time without ever going over the speed limit.

For simplicity the track is divided into section units, each one of them with a specific speed limit. At start position your car marks speed 0 km/h. You can increase your speed or decrease it by multiples of 10. For each 10 km/h your car advances 1 unit. For example, if you have a current speed of 50 km/h your car advances 5 units making what we call a move. After each move, you can change again the car speed. You can for instance accelerate to 70 km/h making your car advance 7 units more, or you can brake to 40 km/h making your car advance 4 units.

While you are running at a determined speed (in a single move), you can only pass track units with speed limit equal or bigger than your current speed. For calculations purposes, a move starts on the unit immediately after the current position. Due to mechanical limitations, rally cars have a maximum acceleration and braking speed. For example a car with a maximum acceleration speed of 30 km/h and maximum break speed of 20 km/h, can only make an increment to the current speed of 30, 20, 10, 0, -10 or -20 km/h.

Your car starts the race before the first unit of the track (a "virtual" zero unit) and to finish the race it must pass over the last unit of the track, passing the finish line. You can cross the finish line at any speed. Note that arriving at the last unit is not considered terminating the race!

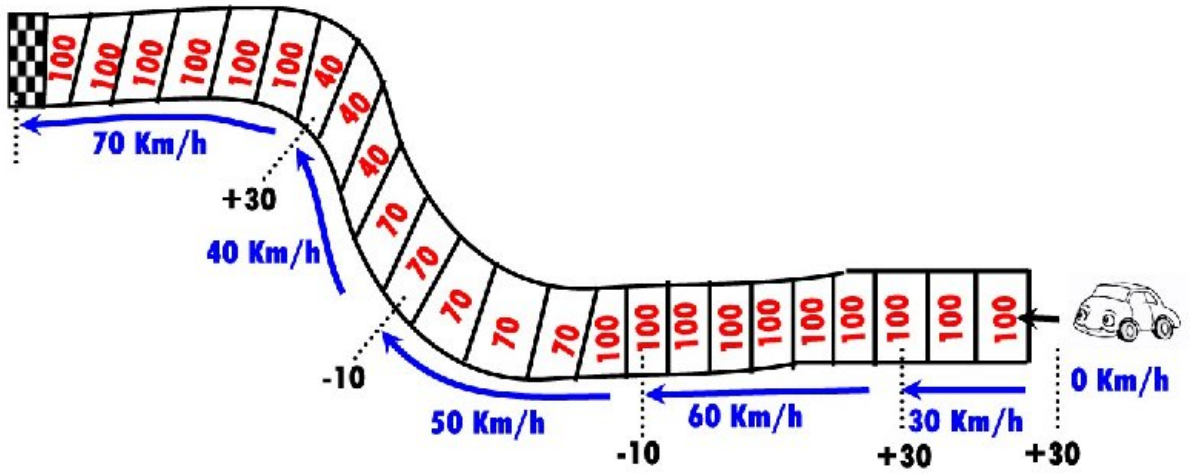
Task

Write a program in Erlang (`rally.erl`), which calculates the number of moves that a car needs to make to finish a track, assuming it uses an optimal strategy, which is one that minimizes the total number of moves, without ever exceeding the speed limit.

Your solution should include tests, including property-based tests, for appropriately chosen parts of the implementation. Your submission must also include a report (`rally.pdf`) explaining your tests.

Example

The following figure illustrates a track example and a corresponding optimal strategy (5 moves). This track is also described in one of the Samples below.



Input and Output

The programs take as input two integers A and B and a list of integer tuples $\{N_i, V_i\}$. Integers A and B indicate the maximum values of acceleration and braking. Each tuple indicates a section of N_i units with speed limit V_i. The end of the track is indicated by the pair $\{0, 0\}$.

Limits

A and B are always multiples of 10, with $10 \leq A, B \leq 240$. V is always a positive multiple of 10 smaller than 250. The maximum number of units in one track is 10.000.

Samples

Here are some sample calls for your program:

```
1> rally:rally(30, 10, [{10,100},{5,70},{3,40},{6,100},{0,0}]).
5
2> rally:rally(40, 50, [{15,100},{0,0}]).
3
3> rally:rally(40, 20, [{1,50},{1,40},{1,30},{1,20},{1,10},
                        {1,20},{1,30},{1,40},{1,50},{0,0}]).
5
```

Testing

You are expected to download and use a tester program before submitting your solutions.

Assuming that you have PropEr in your Erlang library path (following the instructions here), you can use the Erlang module `rallyomatic.beam` (link to download) to test your implementation. Simply put the module in your Erlang code path (e.g., by moving them to your `.beam` directory) and use the exported `proper/0,1` (the second accepting any option `proper:quickcheck/2` would accept in its second argument).

```
$ ls
rally.beam rallyomatic.beam
$ erl
[...]
1> rallyomatic:proper().
.....................................................................
OK: Passed 100 test(s).
true
```