First assignment (2022/2023, this is the one you need to do, due 05-12-2022)
Speed run (part 1)

A road is subdivided into road segments of approximately the same length. Each road segment has a speed limit. The speed is measured by the number of road segments a car is able to advance in a single "move." In each move the car can i) reduce its speed by one (brake), ii) maintain the speed (cruise), or iii) increase its speed by one (accelerate). The car is placed at the first segment of the road with a speed of zero. It has to reach the last segment of the road with a speed of one (at which point it can reduce the speed to zero and so, stop). The purpose of this assignment is to determine the **minimum** number of moves required to reach the final position.

The car position, i.e., the road segment number, is stored in the integer variable `position`. Its final position is stored in `final_position`, and its speed in `speed`. A car move consists in doing the following:

1. Choose its `new_speed`. It can be `speed-1`, `speed`, or `speed+1`. The new speed must be positive.
2. Advance to the new position: `new_position=position+new_speed`.
3. **However**, it can only do so if it never exceeds the speed limit, i.e., for `i=0,1,...,new_speed`, it must be true that `new_speed<=max_road_speed[position+i]`.

In the following example the road has 31 segments (i.e. `final_position=30`).

5	5	6	6	8	7	8	9	8	9	8	9	9	8	9	7	7	6	6	6	5	6	5	4	4	3	5	3	3	3	3
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[25]	[26]	[27]	[28]	[29]	[30]

The optimal solution has 10 moves (the 11 car positions are shown in gray):

5	5	6	6	8	7	8	9	8	9	8	9	9	8	9	7	7	6	6	6	5	6	5	4	4	3	5	3	3	3	3
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[25]	[26]	[27]	[28]	[29]	[30]

Speed run (part 3, what has to be done)

Proposed tasks:

- See how large the final position can be using a `_time_limit_` of 3600.0 (one hour). This has to be done using each one of the student numbers of the group as a (command line) input to the program. Record the execution times as a function of the final position. Try to find a formula that gives a reasonably good estimate of the execution time (as a function of the final position). For a final position of 800, estimate how long would the program take to give an answer.
- As it would be great to be able to reach a `final_position` of 800, which is not possible using the given solution, **invent** other solution methods. Do not change the `solution_1_recursion` and `solve_1` functions. Instead, create new ones. [Hint: it is possible to solve the problem with a final position of 800 in a few microseconds.] Which of your methods is the fastest?
  Consider solving the problem using dynamic programming (that will be explained later in this course). That is not mandatory, but should be attempted by groups with very good students (or so they believe themselves to be). A warning, though. The teachers will give no support regarding using dynamic programming at this early stage. You will be on your own. Impress us!