Specification

Classes

Freeway

Method	Description
`Freeway(capacities, ready_percent, pass_limit)`	Prints `Αυτοκινητόδρομος σε λειτουργία` and constructs its data members
`Operate()`	`void` Calls the `Operate()` method of each `Segment` from end to start and prints the number of `Car` on the `Freeway`
`num_cars()`	`size_t`
`segments()`	`vector<Segment*>`

Segment

Variable		Description
`kMaxCars`	`size_t`	Max number of `Cars` generated in each `Segment` `>= 1`

Method	Description
`Segment(capacity, prev, ready_percent, num_junctions, pass_limit)`	Creates a random number of `Cars`
`Enter()`	`void` Max possible `Cars` enter from `Tolls` and previous `Segment`. Required messages are printed
`Exit()`	`void` `Cars` whose destination is the next junction exit the `Freeway`
`Operate()`	`void` Calls `Exit()`, `Enter()` and randomly sets `ready_percent`% `Cars` as `ready`
`Pass(size_t)`	`void` Max possible `Cars` exit the `Segment` and enter the next one
`cars()`	`vector<Car*>`
`num_cars()`	`size_t`
`ready_cars()`	`vector<Car*>`
`capacity()`	`size_t`
`entrance()`	`size_t`
`set_next(next)`	`void`

Junction

Variable		Description
`kMaxTollsPerType`	`size_t`	Max number of tolls generated in each `Junction` `>= 1`
`kMaxCarsPerToll`	`size_t`	Max number of cars generated in each `Toll` `>= 1`

Method	Description
`Junction(num_junctions, pass_limit)`	Creates a random number of `Tolls`
`Cars()`	`vector<Car*>`
`NumCars()`	`size_t`

Method	Description
`Operate(max_allowed_cars)`	`vector<Car*>` Returns max `Cars` respecting the `Segment.capacity()` and the `pass_limit`. If less than `3 * pass_limit` `Cars` are allowed to enter, the `pass_limit` is decreased. If `3 * pass_limit` `Cars` enter, then the `pass_limit` is increased. Finally, new `Cars` are added in each `Toll`
`current_id()`	`static size_t` Total number of `Junctions` initialized at `0`
`id()`	`size_t`
`pass_limit()`	`size_t`

Toll

Variable	Description
`kMaxCars`	`size_t` Max number of cars generated in each `Toll` `>= 1`

Method	Description
`Toll(current_junction, num_junctions)`	Creates a random number of `Cars`
`Add(car)`	`void`
`Remove()`	`vector<Car*>` Removes all `Cars`
`Remove(num_cars)`	`vector<Car*>` Removes at most `num_cars` `Cars`
`cars()`	`vector<Car*>`
`num_cars()`	`size_t`

Car

Method	Description
`Car(exit_junction, segment)`	`ready` becomes `false`
`exit()`	`size_t`
`set_ready(ready)`	`void`
`ready()`	`bool`
`set_segment(segment)`	`void`
`segment()`	`Segment*`

Usage

The executable file, e.g. `build/project.out`, receives from the command-line the following **case-insensitive** arguments with single, double, or no – prefix:

Argument	Description
`seed`	`uint` Randomness seed
`N`	`int` Simulation steps number
`NSegs`	`size_t` Freeway segments number
`K`	`size_t` Initial max car number that can pass a manned toll station
`Percent`	`int` Car percent on a segment that becomes `ready` in the next step

If any of these arguments is not provided, a default value **must** be used.

During the execution, `NSegs` numbers (`size_t`) are read from the standard input corresponding to the `capacity` of each `Segment`.

E.g:

```
oop-project git:master □ ./build/project.out -n 10 -nsegs 5 -k 10 -percent 30
Seed: 1454857303
N: 10
NSegs: 5
K: 10
Percent: 30
Enter the capacities: 10 15 5 12 17
```

An instance of `Freeway` should be constructed given the above data and then the `Operate()` method should be called `N` times.