Exercise 2 - Requirements and Guidelines

In this exercise you are required to create simulation program that will run several algorithms on several houses.
Submissions shall include 3 different algorithms, as .so files that would be dynamically loaded by the simulation, as would be described below, however, in this exercise the quality of the algorithm would not be reviewed, the only requirements for the three different algorithms are:

(a) to actually behave differently
(b) to be completely deterministic, i.e. not to use any random function and to have the same exact behavior each time it runs on the same house
(c) to have some logic for returning back to the docking station on time (it is ok if algorithm is a bit adventurer and tries to get back to the docking station in a different path, having the risk of not being able to be back on time)

The algorithms that you submit MUST be called after the ID of one of the team members:
<ID>_A_.so with class name for the algorithm = _<ID>_A
<ID>_B_.so with class name for the algorithm = _<ID>_B
<ID>_C_.so with class name for the algorithm = _<ID>_C
All three files must use the same team member ID.

House file Change:
The format of the house file is redefined:
Line 1: house name / description - internal for the file, will NOT be printed in the results
Line 2: MaxSteps for the simulation on this house. This is a new variable in the file (not mentioned in the previous version of the file). We add this variable since we assume different houses may require different MaxSteps. This variable comes instead of the variable with the same name that was part of the configuration.
Line 3: Number of Rows in house
Line 4: Number of Cols in house
Lines 5 and on: the house itself, as described in ex 1

The actual size of the house is determined by the values in lines 3 and 4. If there are missing lines, they should be filled in with empty lines, if there are missing columns they should be filled in with spaces. After filling all the missing characters, the program shall complete the frame of the house with walls.
In case the file contains additional rows and/or cols beyond the required number as set in lines 3 and 4, program shall ignore the redundant lines / cols.
The house validity rules remain the same as in ex 1 (except for a note in p.4 regarding multiple docking stations that are overridden by frame and house is considered valid).

Example of a valid house:

```
Small house
10
4
10
WWWW
W D
W 123456789
```

The house “as should appear in memory”:

```
WWWWWWWWWW
W D      W
W 1234567W
WWWWWWWWWW
```

Program shall get the following command-line arguments:
(a) -config <config path>
(b) -house_path <house path>
(c) -algorithm_path <algorithm path>

Program usage printout is:

```
Usage: simulator [-config <config path>] [-house_path <house path>]
[-algorithm_path <algorithm path>]
```

Note:
Order of arguments is not defined, any order should be valid and supported.
Each of the arguments can be set with absolute or relative path. One argument can be relative and another absolute, or any other combination. Each of the arguments, if provided, shall point to a directory (NOT to a file) and may or may not have a trailing slash.
Each of the arguments may be missing, in which case application should look for the files associated with this argument in the working directory. Other arguments may be present and if so shall be used properly.
Files to look for:

- Config file: config.ini
- House files, files with suffix: .house
- Algorithm files: *_.so

Config
There are no “default” config values. If the file cannot be loaded for some reason or some values are missing in the config file, there is no default values.
In case a config.ini cannot be found in the relevant folder in which we look for the config file, a usage should be printed and the program shall return.

In case the config file exists but cannot be opened or cannot be read or there are missing values, program shall return after printing one of the following error messages, as relevant:
config.ini exists in '<full path>' but cannot be opened
config.ini missing <number> parameter(s): <list of missing parameters>
(If there is a problem with the config file, we do not continue to check algorithms and houses).

Your submission shall include the following config.ini:
(order of parameters shall not be assumed)
MaxStepsAfterWinner=200
BatteryCapacity=400
BatteryConsumptionRate=1
BatteryRechargeRate=20

Algorithms
In case the folder for the algorithm files is bad for some reason or leads to a directory with no algorithm files, or is missing and there are no algorithm files in the working directory - a usage would be printed and the program shall return.
In case there are algorithm files in the relevant directory, but **all** algorithm files are bad or invalid (cannot be opened, malformed, invalid etc.) program shall return after printing error message.
(If there is a problem with algorithm files, we do not continue to check houses).
Error message:
All algorithm files in target folder '<full path of target folder>' cannot be opened or are invalid:
<filename 1>: <problem>
…

The <problem> print shall be one of the following:
file cannot be loaded or is not a valid .so
valid .so but no algorithm was registered after loading it

Houses
In case the folder for the house files is bad for some reason or leads to a directory with no house files, or is missing and there are no house files in the working directory - a usage would be printed and the program shall return.
In case there are house files in the relevant directory, but **all** house files are bad or invalid (cannot be opened, malformed, invalid etc.) program shall return after printing error message:
All house files in target folder '<full path of target folder>' cannot be opened or are invalid:
<filename 1>: <problem>
…

The <problem> print shall be one of the following:
missing docking station (no D in house)
too many docking stations (more than one D in house)
line number <X> in house file shall be a positive number, found: <Y>
cannot open file

Note:
In the case where there are X>1 docking stations in the house and (X-1) are on the outer frame: in such a case we will treat the house as valid house (having only 1 docking after completing house frame with walls). **You should NOT treat such house as invalid house**!

Results
In case there is at least one valid house and one valid algorithm, the immediate first printout would be the simulation results table.

The structure of the results table would be as following:

```
|< 13 chars  >|<10 chars>|<10 chars>|<10 chars>|<10 chars>|...
```

Number of columns = num houses + 1 (first column for algorithm list)
Number of rows = num algorithms + 1 (top row for house list)
Score per each algorithm-house shall be aligned to the right
Spaces shall be exactly kept to create aligned table
Algorithm name = name of the so file without the .so suffix, aligned to left
House name = name of the house file without the .house suffix, trimmed to 9 chars, aligned to left
First line, last line and each line between algorithms would be a line of dashes
Last results column would be the average, as a floating point number with with 2 decimal points, title for this column would be AVG.

Printout example:

```
-------------------------------------------------------------
|             |001       |002       |trimmed_n |AVG       |
-------------------------------------------------------------
|331332334_A_ |       100|       109|      1100|    436.33|
-------------------------------------------------------------
|331332334_B_ |       600|       705|       102|    469.00|
-------------------------------------------------------------
|331332334_C_ |         0|         0|      1500|    500.00|
-------------------------------------------------------------
```

NOTE: do not assume that simulation runs only on 3 algorithms / 3 houses, there can be any number of houses / algorithms.
Order of algorithms (top to bottom) and of houses (left to right) shall be based on case-sensitive order (lexical sort). Yet, it doesn't mean that you need to actually "sort" things, it might be a direct result of the order you traverse through the files.

Errors
If there were no errors, after printing the results table the program shall return.

If there were errors, an empty single new line would separate the results table from the error list.
The error list would start with the word “Errors:” and a single new line after which the errors would be listed.

Errors may include the following:

Houses
In case there were houses which we couldn’t run (problem with the house file), those would not appear in the results table. An error message for the bad houses would appear in the error list, sorted by the house name. The error would be in the format:
<filename 1>: <problem>
Problem messages for houses would be as described above.

Algorithms
In case there were algorithms which we couldn’t run (problem with the algorithm file), those would not appear in the results table. An error message for the bad algorithms would appear in the error list, sorted by the algorithm name. The error would be in the format:
<filename 1>: <problem>
Problem messages for algorithms would be as described above.

Algorithm errors would appear right after the house errors, with no separation.
Each error shall have its own line.
If there are no house errors and there are algorithm errors, those will appear right after the “Errors:” title.

Next page demonstrates some additional printouts of possible runs.

Printout example (run with errors):

```
-------------------------------------------------------------
|             |001       |002       |trimmed_n |AVG       |
-------------------------------------------------------------
|331332334_A_ |       100|       109|      1100|    436.33|
-------------------------------------------------------------
|331332334_B_ |       600|       705|       102|    469.00|
-------------------------------------------------------------
|331332334_D_ |         0|         0|      1500|    500.00|
-------------------------------------------------------------

Errors:
003.house: missing docking station (no D in house)
004.house: too many docking stations (more than one D in house)
005.house: line number 2 in house file shall be a positive number, found: bla
006.house: cannot open file
331332334_C_.so: file cannot be loaded or is not a valid .so
331332334_E_.so: valid .so but no algorithm was registered after loading it
```

Printout example (config file found, but missing one value):

```
config.ini missing 1 parameter(s): BatteryCapacity
```

Printout example (config file found, but missing few values):

```
config.ini missing 2 parameter(s): BatteryCapacity, BatteryRechargeRate
```

Note 1: the order of the missing parameters in the error message is not important, but all the missing parameters shall be listed.
Note 2: additional parameters in config.ini that are not defined shall be ignored, it's not an error.
Note 3: order of parameters in config.ini shall NOT be assumed! parameters may appear in any order.
Note 4: in case a parameter appears in config.ini more than once, it's not an error, the last value shall be taken.

Printout example (no algorithm file found in the relevant directory):

```
Usage: simulator [-config <config path>] [-house_path <house path>]
[-algorithm_path <algorithm path>]
```

Printout example (all algorithm files found in the relevant directory are invalid):

```
All algorithm files in target folder '/si' cannot be opened or are invalid:
331332334_A_.so: file cannot be loaded or is not a valid .so
331332334_B_.so: valid .so but no algorithm was registered after loading it
331332334_C_.so: valid .so but no algorithm was registered after loading it
```