SCALE FOR PROJECT BSQ (/PROJECTS/BOOTCAMP-BSQ)

Guidelines

Before inspecting and grading this project, make sure to read
and apply preliminaries.

Attachments

Subject (https://cdn.intra.42.fr/pdf/pdf/701/1c5de3d2208ff81f724f886bb7e5b6d5-bsq.en.pdf)

Preliminaries

If cheating is suspected, the evaluation stops here. Tick the "Cheat" correction box at the top of this Scale page. You then have to let us know about it by opening a ticket on meta.intrav2.42.fr. In this ticket, you should indicate the Bootcamp day as well as the cheating student's login. => FYI : cheating means the student who is suspected cannot explain of recreate their own code, or he's used elements specifically forbidden in the subject or that go against school policies. Please do this calmy, wisely and with caution.

Simple preliminaries

- Missing team members : 0, evaluation stops.
- Nothing submitted (or wrong file or directory) : 0, evaluation is over.
- The submitted material can only use notions learnt during the bootcamp (No C99, etc), otherwise : 0, evaluation is over.
- The submitted directory must have an author file at its root, with the appropriate format (see subject), otherwise : 0, evaluation is over.
- If there's a Makefile make sure it does compile the project, otherwise : 0, evaluation is over.

La Norme :
- Check the Norm with the help of Norminator. Should there be any norm error, evaluation is over.

⊘ Yes   ✕ No

The subject's pre-requisites

If one of these conditions hasn't been met, the group gets 0, the evaluation stops immediately. Be as intransigent as the moulinette.

- Make sure the Makefile is at the root of the submission directory. Not in ex00/, sources/, or other shenanigans.
- Test compilation (still at the root of the directory) with 'make clean && make && make fclean && make && make clean && ls -la ./bsq && echo OK'. It should display OK at the end, and there should be an executable called "bsq".
- Check whether the program works with 1 to N-number of files as arguments (Test with empty maps of 1x1 for example)
- Check that the program can work without an argument, but is able to read a map from the standard input (./bsq < map, or cat map | ./bsq)
- Check whether the program detects errors within the map's header, as well as within the map itself, and displays the appropriate error message.

If all of these requirements are met, proceed with the rest of the evaluation.

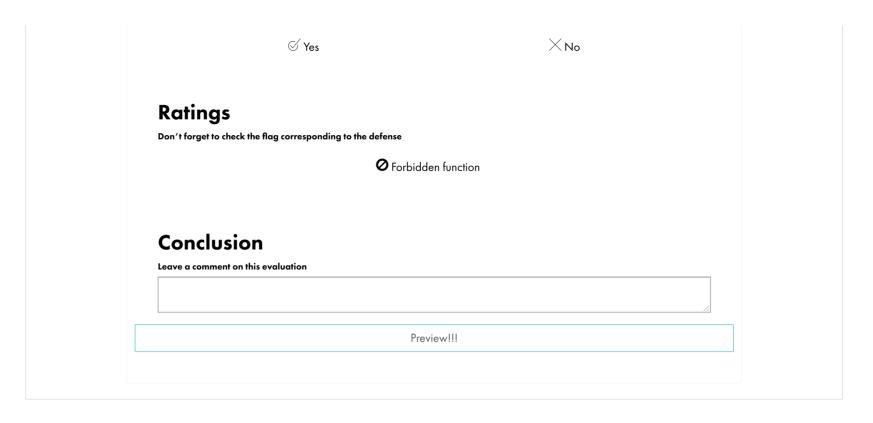⊘ Yes   ✕ No

Feature Tests

You must test the program with your own maps. If a test fails, evaluation stops. Therefore you must run tests in the approriate order and stop as soon as you come accross an error.

Test 00

A 1x1 map that can be filled.

⊘ Yes   ✕ No

Test 01

A 1x1 map that cannot be filled.

⊘ Yes   ✕ No

Test 02

A 5x5 map.

⊘ Yes   ✕ No

Test 03

A chessboard-type map (8x8 1 out of 2 obstacles in diagonal), the character 'full' is indeed the highest and on the far left.

☑ Yes   ✕ No

Test 04

A 10x10 map.

☑ Yes   ✕ No

Test 05

A 10x10 map with a charset other than .oX (For example 123).

☑ Yes   ✕ No

Test 06

A 100x100 map.

☑ Yes   ✕ No

Test 07

A 100x100 map with obstacles.

☑ Yes   ✕ No

Test 08

An empty 100x100 map.

☑ Yes   ✕ No

Test 09

A 100x100 map with multiple large solutions (at least 5x5 per solution), the one on the top far left selected.

☑ Yes   ✕ No

Test 10

A 1000x1000 map.

✓ Yes   ✕ No

Ratings

Don't forget to check the flag corresponding to the defense

⊘ Forbidden function

Conclusion

Leave a comment on this evaluation

Preview!!!