# SCALE FOR PROJECTLEM_IN (/PROJECTS/LEM_IN)

You should correct 2 students in this team

Git repository

## Introduction

Please respect the following rules:

- Remain polite, courteous, respectful and constructive throughout the correction process. The well-being of the community
depends on it.

- Identify with the person (or the group) graded the eventual
dysfunctions of the work. Take the time to discuss
and debate the problems you have identified.

- You must consider that there might be some difference in how your
peers might have understood the project's instructions and the
scope of its functionalities. Always keep an open mind and grade
him/her as honestly as possible. The pedagogy is valid only if
the peer-evaluation is conducted seriously.

## Guidelines

- Only grade the work that is in the student or group'sGiT repository.

- Double-check that the GiT repository belongs to thestudent
or the group. Ensure that the work is for the relevant project and also check that "git clone" is used in an empty folder.

- Check carefully that no malicious aliases was used tofool you
and make you evaluate something other than the content
of the official
repository.

- To avoid any surprises, carefully check that both thecorrecting and the corrected students have reviewed the possible scripts used to facilitate the grading.

- If the correcting student has not completed that particularproject yet, it is mandatory for this student to read the entire subject prior to starting the defence.

- Use the flags available on this scale to signal an emptyrepository, non-functioning program, a norm error, cheating etc. In these cases, the grading is over and the final grade is 0 (or -42 in case of
cheating). However, with the exception of cheating, you are encouraged to continue to discuss your work (even if you have not finished it) in order to identify any issues that may have caused
this failure and avoid repeating the same mistake in the future.

# Attachments

☐ Subject

(https://cdn.intra.42.fr/pdf/pdf/185/lem-in.fr.pdf)

☐ Subiect

(https://cdn.intra.42.fr/pdf/pdf/452/lemin.ro.pdf)

☐ Subject (https://cdn.intra.42.fr/pdf/pdf/947/lemin.en.pdf)

# Mandatory part

*Reminder : Remember that for the duration of the defence, no segfault, nor other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag. This rule is active thoughout the whole defence.*

## Author file

Check that the author file is at the root of the repository
and formatted as explained in the subject. If not the defence is finished and final
grade is 0.

☐ Yes ☐ No

## Reading on the standard output

The program must be able to read an ant-farm on the standard output.
(For example: "./lem-in < ant_farm.txt")

If not, the project is off-topic, the defence is finished and
the final grade
is 0.

☐ Yes ☐ No

# Error management

*The program must manage errors properly on the standard output. Remain open minded when considering the design implemented for that section, and most importantly with regards to the error messages. Don't be picky just make sure it's consistant. Basically, we mean that, if the project displays either something different than ``ERROR'' or if the choice was made to display nothing as an answer when the ant-farm has no ants, you can accept it.*

## No rooms

Evaluate the implementation of error management when there is no room.

☐ Yes ☐ No

## No ants

Evaluate the implementation of error management when there is no ant.

☐ Yes ☐ No

## No mandatory comments

Evaluate the implementation of error management when there is no ##start / ##end
room.

☐ Yes ☐ No

## No possible solution

Evaluate the implementation of error management when there is no possible
solution with the given ant-farm.

☐ Yes ☐ No

# Comment management

## Comments

The program accepts ant-farms with comments.

☐ Yes ☐ No

## Commands

The program accepts ant-farms with commands other than ##start ##end.

☐ Yes ☐ No

# Output format

*The output format must be observed.*

## Ant-farm composition.

The ant-farm's composition is displayed on the standard output.

☐ Yes ☐ No

## Commands and comments

Commands and comments are printed on the standard output.

☐ Yes ☐ No

## Ants movements

The movements are printed with the right format:

- 1 line per turn
- N movements per turn
- A movement is defined as follow "Lx-y" where x is the ant's
number and y a room's name

☐ Yes ☐ No

# Algorithm's accuracy

## The path is valid

In this section, we'll evaluate the algorithm's accuracy and make sure
that the solution works, we're not checking the efficiancy of the program
(this will be covered in the bonus part). It's up to you to make enough
tests with valid ant-farms and check that the output is consistent with
the input you used.

Execute the following 3 tests:

- The solution is making the ants properly go from ##start to ##end.

- Ants aren't walking on each other (one ant per room at a specific
moment in time)

- At the end all the ants are in the ##end room.

Run the same tests several times to make sure the output is always right.

If at least one fails, no points will be awarded for this section. The
defence is finished

☐ Yes ☐ No

# Bonus

## Bonuses

In this section you can evaluate up to 5 operational and well
implemented bonuses.

Bonus example:
- A very efficient algorithm
- Graphic visualizer -> if you find it very well made, you can
give more than 1 point for it.
- ...

**Rate it from 0 (failed) through 5 (excellent)**

# Ratings

**Don’t forget to check the flag corresponding to the defense**

☐ Ok ☐ Outstanding project

☐ Empty work ☐ Incomplete work

☐ No author file ☐ Invalid compilation

☐ Norme ☐ Cheat ☐ Crash

☐ Leaks

# Conclusion

**Leave a comment on this evaluation**

**Finish evaluation**