



(<https://profile.intra.42.fr>)

(<https://profile.intra.42.fr/searches>)  
**SCALE FOR PROJECT RUSHES**  
**(/PROJECTS/RUSHES) / ALCU**  
**(/PROJECTS/RUSHES-ALCU)**

You should correct 3 students in this team



Git repository

`vogsphere@vogsphere-2.unit.ua:intra/2018/activities/rushes_`

## Introduction

We ask you, for a pleasant and useful defense session, to :

- Be polite, respectful and constructive.
- Accept that there can be differences between the way you interpreted the subject and the way others did. Be open-minded about the others' interpretation, anything that is not written black-on-white in the subject can be discussed.

## Guidelines

REMEMBER THAT YOU SHOULD ONLY EVALUATE THE WORK THAT IS ON THE STUDENTS' GIT REPOSITORY.

You must do a "git clone" of their repository, and evaluate what you find there.

## Attachments

Subject (<https://cdn.intra.42.fr/pdf/pdf/658/alum1.en.pdf>)

## Preliminaries

### Preliminaries

First, you will check the following points. Each of these points is mandatory :

- There is a well-formated author file
- The Makefile (with all the usual rules) compiles a program called alum1
- Only authorized functions are used

- Not a single Norm error

If any of those points is not valid, the defense stops. You can still discuss the project, but there will be no points given.

☒ Yes

☐ No

---

## The specs

---

### Launch modes

The program reads the board configuration from the text file given as argument, and defaults to STDIN if no argument is given to the program.

☒ Yes

☐ No

---

### Error handling

You will check that the program handles errors gracefully.

Test it with badly formatted lines (more than 10000 matches per line, more than one number per line, other ASCII characters in a line, empty lines, ...), or with infinite input (/dev/random is your friend for such tests). The program should output "ERROR" on standard error, and exit gracefully.

Of course, error handling should work perfectly whether you are reading the board configuration from a file or from STDIN.

Any error that makes the program actually crash (Segmentation fault, bus error, etc...) ends the defense : if it happens you must select CRASH at the top of the scale.

☒ Yes

☐ No

---

### Memory allocation

Ensure that any memory dynamically allocated is freed properly before the termination of the program.

☒ Yes

☐ No

---

### Game start

So you start a game. The board must be displayed after each turn, be it yours or the computer's. The program should either decide itself who starts the game, or ask the player - whichever is fine.

☒ Yes

☐ No

## Game progression

The program must prompt you for your move, and if your input is invalid (either because it is not between 1 and 3, or because there are not enough matches on the last row), it should prompt you again until you give a good answer.

☒ Yes☐ No

## Game over

At the end of the game, the winner should be announced.

☒ Yes☐ No

## The IA

Play with different configurations, and make sure that the IA tries to win.

There should be a decent algorithm behind the IA : playing randomly or doing the same move over and over again don't count !

In board configurations obviously favoring the IA, it should win.

☒ Yes☐ No

# Bonus

## Bonuses !

You can count up to five bonus points here. It's up to you to judge if the presented bonuses are relevant / useful.

Here are some examples :

- A centered board output (like in the pdf)
- A multiplayer mode (game must still be playable against an IA)
- etc...





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

1

# Ratings

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

☒ Ok

 Empty work Incomplete work No author file Invalid compilation Norme Cheat Crash Incomplete group

## Conclusion

Leave a comment on this evaluation

**Finish evaluation**