SCALE FOR PROJECT C PISCINE RUSH 00

Introduction

Please respect the following rules:

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

- Identify with the person (or the group) evaluated 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 and only if peer-evaluation is conducted seriously.

Guidelines

- Only grade the work that is in the candidate or group's GiT repository.

- Double-check that the GiT repository belongs to the candidate 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 to fool you and make you evaluate something other than the content of the official repository.

- To avoid any surprises, carefully check that both the evaluating and the evaluated candidates have reviewed the possible scripts used to facilitate the grading.

- If the evaluating candidate has not completed that particular project yet, it is mandatory for this candidate to read the entire subject prior to starting the defence.

- Use the flags available on this scale to signal an empty repository, 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.

- Remember that for the duration of the defence, no segfault or bus error can happen, else the final grade is 0.
Use the appropriate flag.

- You should never have to edit any file except the configuration file if it exists.
If you want to edit a file, take the time to explicit the reasons with the evaluated candidate and make sure both of you are okay with this.

- Check that there are only the requested files available in the git repository.
If not, the evaluation stop here.

Attachments

subject.pdf groups_subject.txt

Attachments

subject.pdf groups_subject.txt

How are you doing?

How are you doing?

Ask the candidates how is the piscine going so far.
Are they progressing on their modules?
Are they blocked at a certain point?
Do they meet interesting people?
Are they having the time of their life?

Yes No

Let's go!

Simple preliminaries I - Submission

- Is there something on the repo?
- Are the files submitted in the correct directory?
- Is there a main.c file containing the main function?
- Is there a ft_putchar.c file containing the ft_putchar function?
- Is there a rush0X.c file containing the functions needed for the algorithm? If there are more files than the ones mentioned above which are used for bonuses, they can be located wherever as long as the file name is correct: rush0X.c. If there this step is not correct the evaluation stops here. You should keep going and discuss the implementation of the code, but it will not be graded.

Yes No

Simple preliminaries II - Compilation

You can compile the program using the appropriate flags: -Wall -Wextra -Werror.
It should not print any error.
If there is an error the evaluation stops here, use the appropiate flag.
You should keep going and discuss the implementation of the code,
but it will not be graded.

Yes No

Simple preliminaries III - Norminette

Run the norminette on all the submitted C files.
Norminette prevails when it comes to files and headers evaluation in C
and only in C (that excludes makefile and shell scripts).

A Norm error means the candidate isn't rigorous enough and
the evaluation stops here. You should keep going and discuss the
implementation of the code, but it will not be graded.

Yes No

Basic test

Run the program testing the examples given on the subject.
Grade the exercise if it works as the subject shows.
To do so, you may have to modify the main and compile again.
If this is the case, make sure the candidates understand why you are doing so.

Any unexpected behavior is considered a crash (segfault, bus error,
infinite loop, or any other kind of crash). If this happens,
you should keep going and discuss the implementation of the code,
but it will not be graded.

Yes No

Next step

Test the program with other simple combinations of small positive integers (different from 0).

Any unexpected behavior is considered a crash (segfault, bus error, infinite loop, or any other kind of crash crash). If this happens, you should keep going and discuss the implementation of the code, but it will not be graded.

Yes | No

Specific cases

Test the program with the values 1 and 2.
The program must always return the expected values.

Any unexpected behavior is considered a crash (segfault, bus error, infinite loop, or any other kind of crash crash). If this happens, you should keep going and discuss the implementation of the code, but it will not be graded.

Yes | No

Generic cases

Here is the main part of the evaluation. Be imaginative, take your time to test multiple values and make sure that the program is always responding as expected.

Any unexpected behavior is considered a crash (segfault, bus error, infinite loop, or any other kind of crash crash). If this happens, you should keep going and discuss the implementation of the code, but it will not be graded.

Yes | No

Error handling

Test here using different combinations to make sure that error handling is done correctly. The program should not crash when given a 0 or a negative number.
Try with one negative value, then swap, try with 2 negative values...
It is accepted:

- The program returns an error message
- The program prints nothing and gives back control

Any unexpected behavior is considered a crash (segfault, bus error, infinite loop, or any other kind of crash crash). If this happens, you should keep going and discuss the implementation of the code, but it will not be graded.

Yes | No

Discussion

Take the time to interact with the group. Make sure that each person of the group talks about the code, and check that everyone understands and can explain it. Help them identify and talk about the problems they encountered during the rush.

How did they organize the work?
Did they develop the algorithm together?
Did they learn new concepts by working with each other?

Yes | No

Bonus

Bonuses

This part can be filled ONLY for those who've collected all points for features and explanations. At the examiner's own discretion.

Example de bonus:

- Fulfillment of an extra subject => 1 grade (so up to 4 max)
- Handling of Argc / Argv => 1 grade

If the candidates do not know what Argc / Argv are, take the time to explain to them what it is, and what it is used for.

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 | Invalid compilation | Norme | Cheat

Crash | Incomplete group | ⊘ Forbidden function

Conclusion

Leave a comment on this evaluation

Preview!!!