SCALE FOR PROJECT C PISCINE RUSH 02

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 student or group's GiT repository.

- Double-check that the GiT repository belongs to the student 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 students have reviewed the possible scripts used to facilitate the grading.

- If the evaluating student has not completed that particular project 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 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, no other unexpected, premature, uncontrolled or unexpected termination of the program, 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 student 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 numbers.dict

Features

Check the Norm for each exercise before running tests. A Norm error means the student isn't rigorous enough. And like any error, it means 0. Norminette prevails when it comes to files or headers evaluation in C and only in C (that excludes makefile and shell script).

Makefile

The Makefile musn't relink.

If the Makefile recompiles when doing two successive 'make', tick No.
If the Makefile doesn't recompile the second time, tick Yes.

✓ Yes ✕ No

Error handling

Make sure there are no unnecessary files in the repository (hidden files, temporary files...)
You have to check the program for potential problems. How does it react when you give an empty string? Some letters which aren't numbers? A negative number? A number too big for an unsigned int ?
How do the program behaves when sending a dictionnary argument? and without it?
During the whole evaluation, make sure that there are no leaks in the program.

✓ Yes ✕ No

Features testing

Play with the program to check for everything.
You should try cases like 0, 000000, 10, 10000, 99999999 to check if the program works for everything.
Reminder : The expected way for the program to work is to use only the keys given in the initial dictionnary. Keys defining
other values can often have unexpected behaviors, which should not be eliminatory.

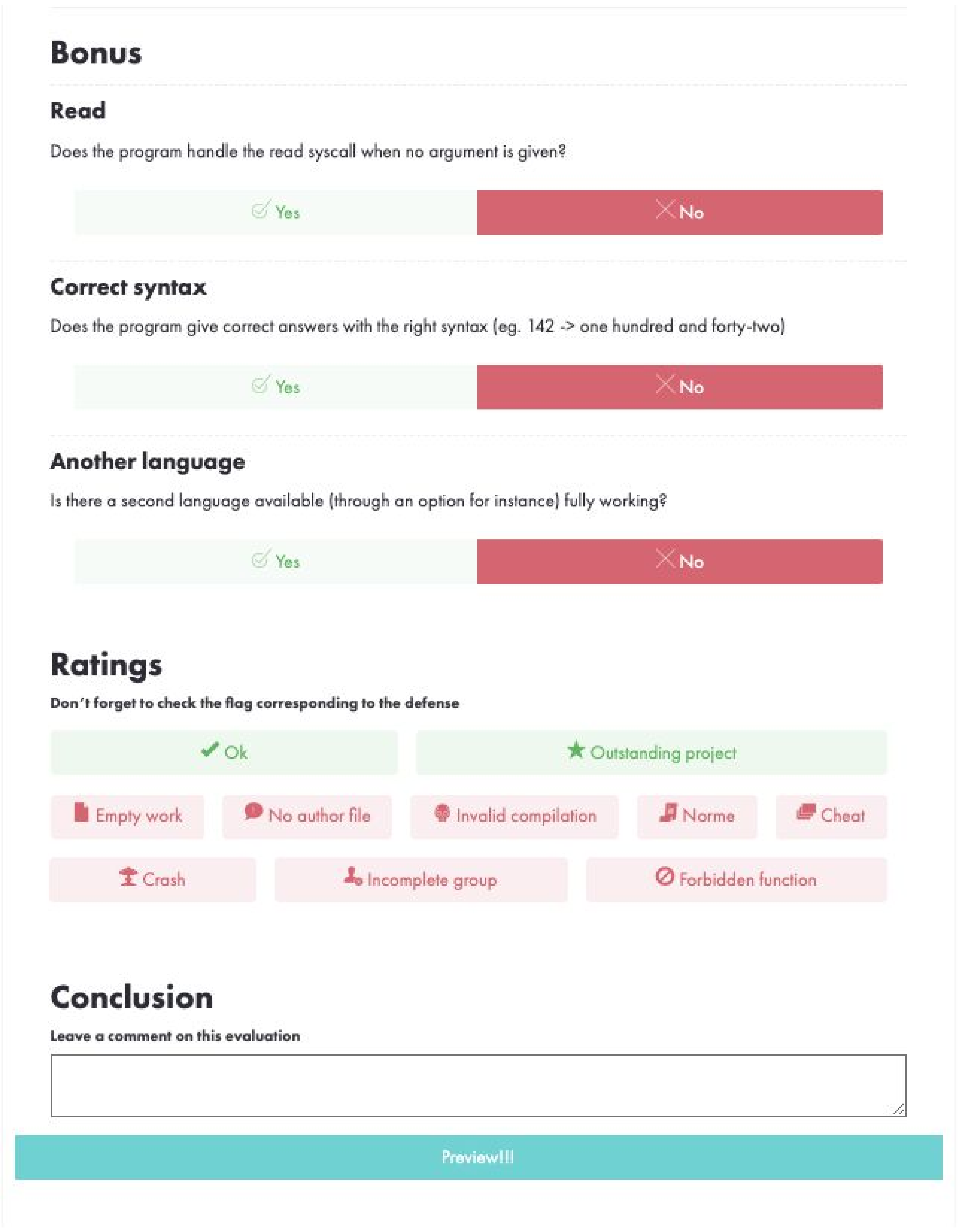
Rate it from 0 (failed) through 5 (excellent)

Full test

Try to find out how the program handles large numbers. Test with very large numbers - thousands, millions, billions... Give points accordingly.

- Expected display up to the thousands. => 2 points
- Expected display up to the millions. => 1 point
- Expected display up to the billions (less than a maximum positive unsigned integer).. => 1 point
- Expected display of the maximum unsigned positive integer. => 1 point

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



Bonus

Read

Does the program handle the read syscall when no argument is given?

Yes [✓ No]

Correct syntax

Does the program give correct answers with the right syntax (eg. 142 -> one hundred and forty-two)

Yes [✓ No]

Another language

Is there a second language available (through an option for instance) fully working?

Yes [✓ No]

Ratings

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

✓ Ok | ★ Outstanding project

Empty work | No author file | Invalid compilation | Norme | Cheat

Crash | Incomplete group | Forbidden function

Conclusion

Leave a comment on this evaluation

Preview!!!