ksonu

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

(https://profile.intra.42.fr/searches)

SCALE FOR PROJECT FILLIT (/PROJECTS/FILLIT)

You should correct 2 students in this team

★

Git repository

```
vogsphere@vgs.42.us.org:intra/2018/activities/fillit/jizheng2
```

Introduction

In order to maintain high evaluation standards, you are expected to:

- Remain polite, courteous, respectful and constructive at every moment
of the discussion. Trust between you and our community depends on
your behaviour.

- Highlight the flaws and issues you uncover in the turned-in
work to the evaluated student or team, and take the time to discuss
every aspect extensively.

- Please take into account that discrepancies regarding the expected
work or functionalities definitions might occur. Keep an open
mind towards the opposite party (is he or she right or wrong?),
and grade as honestly as possible. 42's pedagogy only makes
sense if peer-evaluations are carried out seriously.

Guidelines

- You must grade only what exists in the GiT repository of the
student or team.

- Remember to check the GiT repository's ownership: is it the
student's or team's repository, and for the right project?

- Check thoroughly that no wicked aliases have been used to trick
you into grading something other than the genuine repository.

- Any script supposed to ease the evaluation provided by one party
must be thoroughly checked be the other party in order to avoid unpleasant
situations.

- If the evaluating student hasn't done the project yet, it is mandatory
that he or she reads it before starting the evaluation.

- Use the available flags on this scale to tag an empty work, a non
functional work, a coding style ("norm") error if applicable,
cheating, and so on. If a flag is set, the grade is 0 (or -42 in
case of cheating). However, cheating case excluded, you are encouraged
to carry on discussing what went wrong, why, and how to address it,
even if the grading itself is over.

Attachments

test1.prm (/uploads/document/document/439/test1.prm.txt)

test7.prm (/uploads/document/document/440/test7.prm.txt)

Subject (https://cdn.intra.42.fr/pdf/pdf/886/fillit.en.pdf)

Subject (https://cdn.intra.42.fr/pdf/pdf/734/fillit.fr.pdf)

Preliminaries

Preliminary rules

Verify the followin points :

- Your project must respect the Norm.
- There's no forbidden functions in the reviewed source code.
- The Makefile must be present and must contain the following rules : all,
clean, fclean and re. They must be fully functionnal and correctly
implemented
- Except fo -Wall, -Werror and -Wextra, there will be no more compilation
flags. If you see an optimisation flag (especially -O1, -O2 or -O3),
it is counted as Cheat.
- The executable must be called 'fillit' and once compiled, it should be
at the root of the directory
- The author file is at the root of the repository
and formatted as explained in the subject.

If one of those points is not respected, this evaluation is over. Set the
correct flag, and finish the evaluation.

Nonetheless, you can continue it, for pedagogical purpose, but no points
should be counted.

✓ Yes ✗ No

Error management

In the section, you will evaluate how the program reacts to incorrect inputs and non-working behaviors

Args management

- Test the program with no parameters. It should display the usage followed
by a new line.

- Test the program with more than one parameter. It should display
the usage followed by a new line.

If one of those points is not respected, this evaluation is over. Set
false to this question, and go to the next one.

✓ Yes ✗ No

Invalid piece

- Test the program with a file containing an invalid piece, like this :

```
$> cat test-failing-single-piece.fillit
...#
..#.
.#..
#...
```

It should display 'error' followed by a new line.

- Repeat the process with a piece too large, too small, not squared, with other characters...
It should display 'error' followed by a new line.

If one of those points is not respected, this evaluation is over. Set
false to this question, and go to the next one.

✓ Yes ✗ No

Invalid file

- Test the program with a incorrectly formatted file, like this :

```
$> cat test-incorrect-file.fillit | cat -e
...#$
...#$
...#$
...#$
$
$
...#$
..##$
...#$
$>
```

It should display 'error' followed by a new line.

- Repeat the process with an empty file, more than 26 pieces...
It should display 'error' followed by a new line.

If one of those points is not respected, this evaluation is over. Set false to this question, and go to the next one.

✓ Yes ✗ No

Le Algorithm

In the section, you will evaluate how the program assemble those little tiny things altogether.

Is everything here, as we want ?

Test your program with this file :

```
$> cat test-easy.fillit | cat -e
#...$
#...$
#...$
#...$
$
....$
....$
..##$
..##$
$>
```

On execution, the program should print this on the standard output:

```
$> ./fillit test-easy.fillit
ABB.
ABB.
A...
A...
$>
```

If you don't have the expected output, the algorithm is wrong and/or badly implemented. If one of those points is not respected, this evaluation is over. Set false to this question, and go to the next one.

✓ Yes ✕ No

Le Speed

In the section, you will evaluate how fast the program compute things. Download the two test files of this scale, it should be useful.

Baby steps

Execute the program for test1.prm, with this command :

```
time ./fillit test1.prm
```

If the result takes more than one second, this evaluation is over. Set false to this question, and go to the next one.

✓ Yes ✕ No

Challenge steps

Execute the program for test7.prm, with this command :

```
time ./fillit test7.prm
```

If the result takes more than 30 seconds, set 0 to the correction
If the result takes between 20 and 30 seconds, set 1 to the correction
If the result takes between 10 and 20 seconds, set 2 to the correction
If the result takes between 5 and 10 seconds, set 3 to the correction
If the result takes between 1 and 5 seconds, set 4 to the correction
If the result takes less than 1 second, set 5 to the correction

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 Incomplete group

Conclusion

Leave a comment on this evaluation

Finish evaluation