



## Introduction

Nous vous demandons pour le bon déroulement de cette évaluation de respecter les règles suivantes :

- Restez courtois, polis, respectueux et constructifs en toutes situations lors de cet échange. Le lien de confiance entre la communauté 42 et vous en dépend.

- Mettez en évidence auprès de la personne (ou du groupe) notée les dysfonctionnements éventuels du travail rendu, et prenez le temps d'en discuter et d'en débattre.

- Acceptez qu'il puisse y avoir parfois des différences d'interprétation sur les demandes du sujet ou l'étendue des fonctionnalités. Restez ouvert d'esprit face à la vision de l'autre (a-t-il ou elle raison ou tort ?), et notez le plus honnêtement possible. La pédagogie de 42 n'a de sens que si la peer-évaluation est faite sérieusement.

## Guidelines

- Vous ne devez évaluer que ce qui se trouve sur le dépôt GiT de rendu de l'étudiant(e) ou du groupe.

- Prenez soin de vérifier que le dépôt GiT est bien celui correspondant à l'étudiant(e) ou au groupe, et au projet.



- Vérifiez méticuleusement qu'aucun alias malicieux n'a été utilisé pour vous induire en erreur et vous faire évaluer autre chose que le contenu du dépôt officiel.


- Tout script sensé faciliter l'évaluation fourni par l'un des deux partis doit être rigoureusement vérifié par l'autre parti pour éviter des mauvaises surprises.

- Si l'étudiant(e) correcteur/correctrice n'a pas encore fait ce projet, il est obligatoire pour cet(te) étudiant(e) de lire le sujet en entier avant de commencer cette soutenance.

- Utilisez les flags disponibles sur ce barème pour signaler un rendu vide, non fonctionnel, une faute de norme, un cas de triche, etc. Dans ce cas, l'évaluation est terminée et la note finale est 0 (ou -42 dans le cas spécial de la triche). Toutefois, hors cas de triche, vous êtes encouragés à continuer d'échanger autour du travail effectué (ou non effectué justement) pour identifier les problèmes ayant entraîné cette situation et les éviter pour le prochain rendu.

## Attachments

 test1 (/uploads/document/document/298/test1.prm)  test7 (/uploads/document/document/299/test7.prm)

 Subject (<https://cdn.intra.42.fr/pdf/pdf/635/fillit.fr.pdf>)

## Sections

### Préliminaires

#### Consignes préliminaires

Vous allez commencer par vérifier les points suivants :

- Le projet doit être à la norme.
- On ne doit pas trouver de fonctions interdites.
- Le Makefile doit être présent et propose les règles suivantes : all, clean, fclean, re. Elles doivent être correctement implémentées.
- On ne doit pas trouver d'autres flags de compilation que -Wall, -Werror, -Wextra.
- L'exécutable s'appelle fillit et doit être à la racine du dépôt.

Si un des points n'est pas respecté, la correction s'arrête. Merci d'utiliser le flag adapté. Vous êtes encouragés à continuer la soutenance, mais les points ne seront pas pris en compte.

✓ Yes

✗ No

## Gestion d'erreur

Dans cette partie nous allons tenter de mettre en défaut votre gestion des erreurs.

### Nombre de paramètres

Testez le programme sans paramètre, avec trop de paramètres.  
Le programme vous envoie bien un message d'erreur ?  
Sinon, répondez non, la correction s'arrête.

Testez le programme avec un fichier valide en paramètre.  
Le programme s'exécute normalement ?  
Sinon, répondez non, la correction s'arrête.

✓ Yes

✗ No

### Pièce invalide

Testez le programme avec une pièce invalide à l'intérieur du fichier, par exemple, celle-ci :

```
...#  
..#.  
.#..  
#...
```

Ou encore une pièce trop grande, trop petite...

Le programme vous envoie bien un message d'erreur ?  
Sinon, répondez non, la correction s'arrête.

✓ Yes

✗ No

### Fichier invalide

Testez le programme avec un fichier mal formaté, par exemple, celle-ci :

```
...#  
...#  
...#  
...#  
*deux lignes vides*  
...#  
..##  
...#
```

Ou encore un fichier vide.

Le programme vous envoie bien un message d'erreur ?  
Sinon, répondez non, la correction s'arrête.

✓ Yes

✗ No

## L'algo

Dans cette partie, nous allons vérifier que vous avez correctement implémenté votre algo.

Tout est là où il faut, comme il faut ?

Testez avec un fichier formaté de telle sorte :

```
#...  
#...  
#...  
#...  
*ligne vide*  
....  
....  
..##  
..##
```

La sortie correspond-elle à ce qui suit ?

```
"###.  
####
```

```
###.  
#...  
#..."
```

Si c'est :

```
"###.  
###.  
..#.  
..#."
```

L'algo ne fonctionne pas, répondez non, la correction s'arrête là.

✓ Yes

✗ No

## Le temps

Dans cette partie, nous allons vérifier que votre algo est performant.

### Test simple

Exécutez le fichier test1.prm présent en ressource en utilisant time :

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

Le résultat prend moins d'une seconde pour s'afficher ?

Sinon, répondez non, la correction s'arrête.

✓ Yes

✗ No

### Test plus avancé

Exécutez le fichier test1.prm présent en ressource en utilisant time :

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

Le résultat prend plus de 30 secondes pour s'afficher => 0

Le résultat prend entre 20 et 30 secondes pour s'afficher => 1

Le résultat prend entre 10 et 20 secondes pour s'afficher => 2

Le résultat prend entre 5 et 10 secondes pour s'afficher => 3

Le résultat prend entre 1 et 5 secondes pour s'afficher => 4

Le résultat prend moins d'une seconde pour s'afficher => 5

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

5

## Conclusion

Leave a comment on this correction

\* (required) Comment

Finish correction