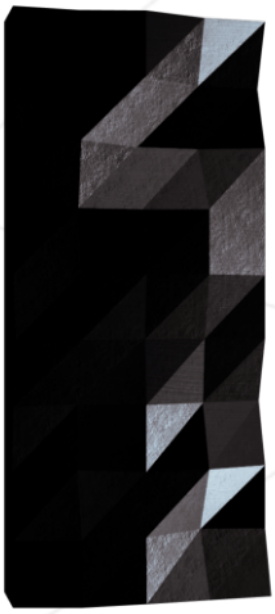

Piscine

05

Résumé: ce document est le sujet du module C 05 de l piscine C de 42.

Version: 6.3

T ble des m tières

Ch pitre I

Consignes

Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.

Relisez bien le sujet avant de rendre vos exercices. tout moment le sujet peut changer.

ttention aux droits de vos fichiers et de vos répertoires.

Vous devez suivre la procédure de rendu pour tous vos exercices.

Vos exercices seront corrigés par vos camarades de piscine.

En plus de vos camarades, vous serez corrigés par un programme appelé la Moulinette.

La Moulinette est très stricte dans sa notation. Elle est totalement automatisée. Il est impossible de discuter de sa note avec elle. Soyez d'une rigueur irréprochable pour éviter les surprises.

La Moulinette n'est pas très ouverte d'esprit. Elle ne cherche pas à comprendre le code qui ne respecte pas la Norme. La Moulinette utilise le programme `norminette` pour vérifier la norme de vos fichiers. Comprendre par là qu'il est stupide de rendre un code qui ne passe pas la `norminette`.

Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.

L'utilisation d'une fonction interdite est un cas de triche. Toute triche est sanctionnée par la note de `-42`.

Vous ne devrez rendre une fonction main() que si nous vous demandons un programme.

La Moulinette compile avec les flags -Wall -Wextra -Werror, et utilise `cc`.

Si votre programme ne compile pas, vous aurez `0`.

Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices.

Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.

Votre manuel de référence s'appelle `Google / m n / Internet / ...`.

Pensez à discuter sur le forum Piscine de votre Intra, ainsi que sur le slack de votre Piscine !

Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...

Réfléchissez. Par pitié, par Odin ! Nom d'une pipe.

`Pour cette journée, la norminette doit être lancée avec le flag -R CheckForbiddenSourceHe der. La moulinette l'utilisera aussi.`

Ch pitre II

Pré mbule

Voici des paroles extraite du premier livre de la saga H rry Potter :

Je n'suis p s d'une be uté suprême
M is f ut p s s'fier à ce qu'on voit
Je veux bien me m nger moi-même
Si vous trouvez plus m lin qu'moi.

Les h uts-d'forme, les ch pe ux splendides,
Font pâl'figure uprès de moi
C r à Poudl rd, qu nd je décide,
Ch cun se soumet à mon choix.

Rien ne m'éch pp' rien ne m' rrête
Le Choixpe u toujours r ison
Mettez-moi donc sur votre tête
Pour conn itre votre m ison.

Si vous llez à Gryffondor
Vous rejoindrez les cour geux,
Les plus h rdis et les plus forts
Sont r ssemblés en ce h ut lieu.

Si à Poufsouffle vous llez,
Comme eux vous s'rez juste et loy l
Ceux de Poufsouffle iment tr v iller
Et leur p tience est proverbi le.

Si vous êtes s ge et réfléchi
Serd igle vous ccueiller peut-être
Là-b s, ce sont des érudits
Qui ont envie de tout conn ître.

Vous finirez à Serpent rd
Si vous êtes plutôt m lin,
C r ceux-là sont de vr is roubl rds
Qui p rviennent toujours à leurs fins.

```
Sur t  tête pose-moi un inst nt
Et n' ie p s peur, reste serein
Tu ser s en de bonnes m ins
C r je suis un ch pe u pens nt !
```

Ce sujet n'a, malheureusement, rien à voir avec la série `H rry Potter`, et c'est dommage, parce que votre rendu ne sera pas fait par `m gie`.

Ch pitre III

Exercice 00 : ft_iter tive_f ctori l

	Exercice : 00
ft_iterative_factorial	
Dossier de rendu : *ex*00	
Fichiers à rendre : `ft_iterative_factorial.c`	
Fonctions utorisées : `ucune`	

Écrire une fonction itérative qui renvoie un nombre. Ce nombre est le résultat de l'opération factorielle à partir du nombre passé en paramètre.

Si l'argument n'est pas valide, la fonction doit renvoyer 0.

Il ne faut pas gerer les "int overflow", le retour de la fonction sera indefini.

Elle devra être prototypée de la façon suivante :

```
int ft_iter tive_f ctori l(int nb);
```

Ch pitre IV

Exercice 01 : ft_recursive_f ctori l

	Exercice : 01
ft_recursive_factorial	
Dossier de rendu : *ex*01	
Fichiers à rendre : `ft_recursive_factorial.c`	
Fonctions utorisées : `ucune`	

Écrire une fonction récursive qui renvoie la factorielle du nombre passé en paramètre.

Si l'argument n'est pas valide, la fonction doit renvoyer 0.

Il ne faut pas gerer les "int overflow", le retour de la fonction sera indefini.

Elle devra être prototypée de la façon suivante :

```
int ft_recursive_f ctori l(int nb);
```

Ch pitre V

Exercice 02 : ft_iter tive_power

	Exercice : 02
ft_iterative_power	
Dossier de rendu : *ex*02	
Fichiers à rendre : `ft_iterative_power.c`	
Fonctions utorisées : `ucune`	

Écrire une fonction itérative qui renvoie une puissance d'un nombre. Une puissance inferieur à 0 renverra 0.

Comme il n'y a pas de concensus sur 0 puissance 0, nous considererons que le resultat sera 1.

Il ne faut pas gerer les "int overflow", le retour de la fonction sera indefini.

Elle devra être prototypée de la façon suivante :

```
int ft_iter tive_power(int nb, int power);
```

Ch pitre VI

Exercice 03 : ft_recursive_power

	Exercice : 03
ft_recursive_power	
Dossier de rendu : *ex*03	
Fichiers à rendre : `ft_recursive_power.c`	
Fonctions utorisées : `ucune`	

Écrire une fonction récursive qui renvoie une puissance d'un nombre.

Comme il n'y a pas de concensus sur 0 puissance 0, nous considererons que le resultat sera 1.

Il ne faut pas gerer les "int overflow", le retour de la fonction sera indefini.

Elle devra être prototypée de la façon suivante :

```
int ft_recursive_power(int nb, int power);
```

Ch pitre VII

Exercice 04 : ft_fibon cci

	Exercice : 04
ft_fibonacci	
Dossier de rendu : *ex*04	
Fichiers à rendre : `ft_fibonacci.c`	
Fonctions utorisées : `ucune`	

Écrire une fonction `ft_fibon cci` qui renvoie le `n`-ième élément de la suite de Fibonacci, le premier élément étant à l'index 0. Nous considererons que la suite de Fibonacci commence par 0, 1, 1, 2.

Les overflows ne devront pas être gerés.

Elle devra être prototypée de la façon suivante :

```
int ft_fibon cci(int index);
```

Évidemment, `ft_fibon cci` devra être récursive.

Si `index` est inférieur à 0, la fonction renverra -1.

Ch pitre VIII

Exercice 05 : ft_sqrt

	Exercice : 05
ft_sqrt	
Dossier de rendu : *ex*05	
Fichiers à rendre : `ft_sqrt.c`	
Fonctions utorisées : `ucune`	

Écrire une fonction qui renvoie la racine carrée entière d'un nombre si elle existe, 0 si la racine carrée n'est pas entière.

Elle devra être prototypée de la façon suivante :

```
int ft_sqrt(int nb);
```

Ch pitre IX

Exercice 06 : ft_is_prime

	Exercice : 06
ft_is_prime	
Dossier de rendu : *ex*06	
Fichiers à rendre : `ft_is_prime.c`	
Fonctions utorisées : `ucune`	

Écrire une fonction qui renvoie 1 si le nombre est premier et 0 si le nombre ne l'est pas.

Elle devra être prototypée de la façon suivante :

```
int ft_is_prime(int nb);
```

`0 et 1 ne sont pas des nombres premiers.`

Ch pitre X

Exercice 07 : ft_find_next_prime

	Exercice : 07
ft_find_next_prime	
Dossier de rendu : *ex*07	
Fichiers à rendre : `ft_find_next_prime.c`	
Fonctions utorisées : `ucune`	

Écrire une fonction qui renvoie le nombre premier immédiatement supérieur ou égal au nombre passé en paramètre.

Elle devra être prototypée de la façon suivante :

```
int ft_find_next_prime(int nb);
```

Ch pitre XI

Exercice 08 : Les dix d mes

	Exercice : 08
Les dix dames	
Dossier de rendu : $ex08$	
Fichiers à rendre : `ft_ten_queens_puzzle.c`	
Fonctions utorisées : `write`	

Écrire une fonction qui affiche toutes les possibilités de placer dix dames sur un échiquier de 10x10 sans qu'elles ne puissent s'atteindre en un seul coup.

La recursivité devra être utilisée.

La valeur de retour de votre fonction devra être le nombre de solutions affichées

Elle devra être prototypée de la façon suivante :

```
int ft_ten_queens_puzzle(voi );
```

L'affichage se fera de la façon suivante :

```
$>./ .out | c t -e
0257948136$
0258693147$
...
4605713829$
4609582731$
...
9742051863$
$>
```

La suite se lit de gauche à droite. Le premier chiffre correspond à la position de la première dame dans la première colonne (l'index commençant à 0). Le énième chiffre correspond à la position de la énième dame dans la énième colonne.

Ch pitre XII

Rendu et peer-ev lu tion

Rendez votre travail sur votre dépot `Git` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.

`Vous ne devez rendre uniquement les fichiers demandés par le sujet de ce projet.`