



C Piscine

C 05

*Summary:* このドキュメントはC Piscine @ 42の C 04モジュール用の課題です。

# Contents

I	Instructions	2
II	Foreword	4
III	Exercise 00 : ft_iterative_factorial	6
IV	Exercise 01 : ft_recursive_factorial	7
V	Exercise 02 : ft_iterative_power	8
VI	Exercise 03 : ft_recursive_power	9
VII	Exercise 04 : ft_fibonacci	10
VIII	Exercise 05 : ft_sqrt	11
IX	Exercise 06 : ft_is_prime	12
X	Exercise 07 : ft_find_next_prime	13
XI	Exercise 08 : The Ten Queens	14

# Chapter I

## Instructions

- このページのみを参考にしてください。噂を信用しないで下さい。
- この書類は、提出前に変更になる可能性があります。十分に注意して下さい。
- ファイルとディレクトリへの権限があることをあらかじめ確認して下さい。
- 課題は全て提出手順に従って行って下さい。
- 課題の確認と評価は、あなたのクラスメイトが行います。
- 課題はMoulinetteと呼ばれるプログラムによっても確認・評価されます。
- Moulinetteは大変細かい評価を行います。全て自動で行われ、交渉方法はありません。頑張ってください。
- Moulinetteは規範を無視したコードは解読できません。Moulinetteはあなたのファイルが規範を遵守しているかをチェックするために、`norminette`と呼ばれるプログラムを使って判断します。要約：せっかくの取り組みが`norminette`のチェックによって無駄になるのは勿体無いので、気をつけましょう。
- 課題は簡単なものから徐々に難しくなるように並べられています。簡単な課題が解けていない場合、難しい問題かが解けていたとしても **加点されることはありません**。
- 禁止されている関数をしようした場合は不正とみなします。不正者は-42の評価をつけられこの評価に交渉の余地はありません。
- プログラムを要求する際は`main()`関数のみを提出しましょう。
- Moulinetteはこれらのフラグを用いてgccでコンパイルします：-Wall -Wextra -Werror。
- プログラムが `コンパイルされなかった場合、評価は0です。
- 課題で指定されているもの以外はどんなファイルもディレクトリ内に残しておくことはできません。
- 質問があれば右側の人に聞きましょう。それでも分からなければ左側の人に聞いてください。

- あなたを助けてくれるのはGoogle / 人間 / インターネット / ...と呼ばれているものです。
- intranet上のフォーラムの” C Piscine” パートかPiscineのslackを確認してください。
- 例を徹底的に調べてください。課題で言及されていない詳細まで要求されます。



Norminetteは、 `-R CheckForbiddenSourceHeader` をオプションに追加しなければなりません。その際、Moulinetteも使用します。

# Chapter II

## Foreword

Here are some lyrics extract from the Harry Potter saga:

Oh you may not think me pretty,  
But don't judge on what you see,  
I'll eat myself if you can find  
A smarter hat than me.

You can keep your bowlers black,  
Your top hats sleek and tall,  
For I'm the Hogwarts Sorting Hat  
And I can cap them all.

The Sorting Hat, stored in the Headmaster's Office.  
There's nothing hidden in your head  
The Sorting Hat can't see,  
So try me on and I will tell you  
Where you ought to be.

You might belong in Gryffindor,  
Where dwell the brave at heart,  
Their daring, nerve, and chivalry  
Set Gryffindors apart;

You might belong in Hufflepuff,  
Where they are just and loyal,  
Those patient Hufflepuffs are true  
And unafraid of toil;

Or yet in wise old Ravenclaw,  
If you've a ready mind,  
Where those of wit and learning,  
Will always find their kind;

Or perhaps in Slytherin  
You'll make your real friends,  
Those cunning folks use any means


To achieve their ends.

So put me on! Don't be afraid!  
And don't get in a flap!  
You're in safe hands (though I have none)  
For I'm a Thinking Cap!

Unfortunately, this subject's got nothing to do with the Harry Potter saga, which is too bad, because your exercises won't be done by magic.

## Chapter III

### Exercise 00 : ft\_iterative\_factorial


	Exercise 00
	ft_iterative_factorial
	提出するディレクトリ : <i>ex00/</i>
	提出するファイル : <i>ft_iterative_factorial.c</i>
	使用可能な関数 : None

- 数字を返す反復関数を作成しましょう。この数字はパラメータとして与えられた数に基づく階乗の結果です。
- 引数が有効でない場合は0を返します。
- オーバーフローは決して処理しないでください。オーバーフローの場合、関数の戻り値が未定義になります。
- プロトタイプ例

```
int ft_iterative_factorial(int nb);
```

# Chapter IV

## Exercise 01 : ft\_recursive\_factorial

	Exercise 01
	ft_recursive_factorial
	提出するディレクトリ : <i>ex01/</i>
	提出するファイル : <i>ft_recursive_factorial.c</i>
	使用可能な関数 : None


- パラメータとして与えられた数の階乗を返す再帰関数を作成しましょう。
- 引数が有効でない場合は0を返します。
- オーバーフローは決して処理しないでください。オーバーフローの場合、関数の戻り値が未定義になります。
- プロトタイプ例

```
int ft_recursive_factorial(int nb);
```



# Chapter V

## Exercise 02 : ft\_iterative\_power


	Exercise 02
	ft_iterative_power
	提出するディレクトリ : <i>ex02/</i>
	提出するファイル : <i>ft_iterative_power.c</i>
	使用可能な関数 : None

- 引数として与えられた数にべき乗の値を返す、反復関数を作成しましょう。  
0未満のべき乗は0を返します。オーバーフローの処理はしないでください。
- 0の0乗は1を返します。
- プロトタイプ例

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

# Chapter VI

## Exercise 03 : ft\_recursive\_power


	Exercise 03
	ft_recursive_power
	提出するディレクトリ : <i>ex03/</i>
	提出するファイル : <i>ft_recursive_power.c</i>
	使用可能な関数 : None

- 引数として与えられた数にべき乗の値を返す再帰関数を作成しましょう。
- オーバーフローは決して処理しないでください。オーバーフローの場合、関数の戻り値が未定義になります。
- 0の0乗は1を返します。
- プロトタイプ例

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

# Chapter VII

## Exercise 04 : ft\_fibonacci

	Exercise 04
	ft_fibonacci
提出するディレクトリ : <i>ex04/</i>	
提出するファイル : <i>ft_fibonacci.c</i>	
使用可能な関数 : None	


- フィボナッチ数列の *n* 番目の要素を返す関数 *ft\_fibonacci* を作成しましょう。最初の要素は0のインデックスにあります。尚、フィボナッチ数列は0、1、1、2のように始まると考えます。
- オーバーフローは決して処理しないでください。オーバーフローの場合、関数の戻り値が未定義になります。
- プロトタイプ例

```
int ft_fibonacci(int index);
```

- *ft\_fibonacci* は必ず再帰的です。
- *index* が0未満の場合、関数は-1を返します。

# Chapter VIII

## Exercise 05 : ft\_sqrt


	Exercise 05
	ft_sqrt
提出するディレクトリ : <i>ex05/</i>	
提出するファイル : <i>ft_sqrt.c</i>	
使用可能な関数 : None	

- （存在する場合は）数値の平方根を返すか、無理数の場合は0を返す関数を作成しましょう。
- プロトタイプ例

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

# Chapter IX

## Exercise 06 : ft\_is\_prime

	Exercise 06
	ft_is_prime
	提出するディレクトリ : <i>ex06/</i>
	提出するファイル : <i>ft_is_prime.c</i>
	使用可能な関数 : None

- パラメータとして与えられた数が素数の場合は1を返し、そうでない場合は0を返す関数を作成しましょう。
- プロトタイプ例


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



0 and 1 are not prime numbers.

# Chapter X

## Exercise 07 : ft\_find\_next\_prime


	Exercise 07
	ft_find_next_prime
	提出するディレクトリ : <i>ex07/</i>
	提出するファイル : <i>ft_find_next_prime.c</i>
	使用可能な関数 : None

- パラメータとして与えられた数より等しいか大きくなる次の素数を返す関数を作成しましょう。
- プロトタイプ例

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

# Chapter XI

## Exercise 08 : The Ten Queens

	Exercise 08
	The Ten Queens
	提出するディレクトリ : <i>ex08/</i>
	提出するファイル : <i>ft_ten_queens_puzzle.c</i>
	使用可能な関数 : <i>write</i>

- 10列× 10行のチェス盤上で、10人のクイーンが一手で接触しない全てのパターン数を返り値として戻す関数を作成しましょう。
- 問題の解決には再帰性が必要です。
- プロトタイプ例

```
int ft_ten_queens_puzzle(void);
```

- 表示例

```
$>./a.out | cat -e
0257948136$
0258693147$
...
4605713829$
4609582731$
...
9742051863$
$>
```

- 列は左から右です。最初の桁は最初の列の最初のクイーンの位置を表します (0から始まるインデックス)。 N桁はN列のN番目のクイーンの位置を表します。
- 返り値は表示結果の合計値でなければなりません。