



C Piscine

C 05

*Summary:* このドキュメントは、*C Piscine @ 42*の *C 05*モジュール用の課題である。

# Contents

|             |   |           |
|-------------|---|-----------|
| <b>I</b>    | <b>Instructions</b>                         | <b>2</b>  |
| <b>II</b>   | <b>Foreword</b>                             | <b>4</b>  |
| <b>III</b>  | <b>Exercise 00 : ft_iterative_factorial</b> | <b>6</b>  |
| <b>IV</b>   | <b>Exercise 01 : ft_recursive_factorial</b> | <b>7</b>  |
| <b>V</b>    | <b>Exercise 02 : ft_iterative_power</b>     | <b>8</b>  |
| <b>VI</b>   | <b>Exercise 03 : ft_recursive_power</b>     | <b>9</b>  |
| <b>VII</b>  | <b>Exercise 04 : ft_fibonacci</b>           | <b>10</b> |
| <b>VIII</b> | <b>Exercise 05 : ft_sqrt</b>                | <b>11</b> |
| <b>IX</b>   | <b>Exercise 06 : ft_is_prime</b>            | <b>12</b> |
| <b>X</b>    | <b>Exercise 07 : ft_find_next_prime</b>     | <b>13</b> |
| <b>XI</b>   | <b>Exercise 08 : The Ten Queens</b>         | <b>14</b> |
| <b>XII</b>  | <b>Submission and peer-evaluation</b>       | <b>15</b> |

# Chapter I

## Instructions

- 課題に関する噂に惑わされないよう気をつけ、信用しないこと。
- この書類は、提出前に変更になる可能性があるため、気をつけること。
- ファイルとディレクトリへの権限があることを、あらかじめ確認すること。
- すべての課題は、提出手順に従い行うこと。
- 課題の確認と評価は、あなたの周りにいるPiscine受験者により行われる。
- 課題の確認と評価は、Piscine受験者に加えて、Moulinetteと呼ばれるプログラムによっても行われる。
- Moulinetteは、大変細かい評価を行う。これはすべて自動で行われるため、交渉の余地はない。
- Moulinetteは、コーディング規範（Norm）を遵守しないコードを解読することができない。そのため、Moulinetteはnorminetteと呼ばれるプログラムを使用し、あなたのファイルがコーディング規範を遵守しているか確認を行う。せっかくの取り組みが、norminetteの確認により無駄にならないよう、気をつけること。
- 問題は、簡単なものから徐々に難しくなるように並べられている。簡単な問題が解けていない場合は、難しい問題が解けていたとしても 加点されることはない。
- 使用が禁止されている関数を使用した場合は、不正とみなされる。不正者は-42の評価をつけられ、この評価に対する交渉の余地はない。
- 課題がプログラムの提出を要求する場合のみ、main()関数を提出すること。
- Moulinetteは以下のフラッグを用いて、ccでコンパイルする。 -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になる。
- 課題で指定されていないものは、どんなファイルもディレクトリ内に置かないこと。

- 質問がある場合は、隣の人に聞くこと。それでも分からない場合は、反対側の席の人に聞くこと。
- 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たちである。
- 出力例には、問題文に明記されていない細部まで表示されている場合があるため、入念に確認すること。



norminetteは、 `-R CheckForbiddenSourceHeader` をオプションに追加しなければならない。Moulinetteも、このオプションを使用する。

# 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  |             |

- 引数として与えられた数の階乗を返す反復関数を作成せよ。
- 引数が無効である場合は、0を返すこと。
- オーバーフローは、処理しないこと。オーバーフローが発生した場合、関数の戻り値は未定義になる。
- 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  |             |


- 引数として与えられた数の階乗を返す再帰関数を作成せよ。
- 引数が無効である場合は、0を返すこと。
- オーバーフローは、処理しないこと。オーバーフローが発生した場合、関数の戻り値は未定義になる。
- 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のべき乗の値を返す反復関数を作成せよ。0未満のべき乗は、0を返すこと。オーバーフローは、処理しないこと。オーバーフローが発生した場合、関数の戻り値は未定義になる。
- 0の0乗は、1を返す。
- 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のべき乗の値を返す再帰関数を作成せよ。
- オーバーフローは、処理しないこと。オーバーフローが発生した場合、関数の戻り値は未定義になる。
- 0の0乗は、1を返すこと。
- 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* を作成せよ。最初の要素は、0番目に配置されている。フィボナッチ数列は、0、1、1、2 のような順序で始まる。
- オーバーフローは、処理しないこと。オーバーフローが発生した場合、関数の戻り値は未定義になる。
- プロトタイプ例)

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

- *ft\_fibonacci* は、再帰的であること。
- *index* が0未満の場合、関数は-1を返す。

# 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  |             |

- 与えられた数に、自然数の平方根がある場合はその値を返し、それ以外の場合は0を返す関数を作成せよ。
- 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  |             |

- 引数として与えられた数が素数である場合は1を返し、それ以外の場合は0を返す関数を作成せよ。
- 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  |             |

- 引数として与えられた数が素数である場合は、同じ数字を返し、それ以外の場合は、その数より大きい次の素数を返す関数を作成せよ。
- 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すべてのパターンの数を返し、全てのパターンを標準出力に出力する関数を作成せよ。
- 問題を解くためには、再帰性が必要となる。
- プロトタイプ例)

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

- 出力例)

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

- 列は左から右に進む。最初の桁は、最初の列にある最初のクイーンの位置を表す。(0から始まるインデックス) N桁は、N列にあるN番目のクイーンの位置を表す。
- 戻り値は、出力結果の合計値であること。

# Chapter XII

## Submission and peer-evaluation

課題は、いつも通り Git リポジトリに提出すること。リポジトリ内の提出物のみが、レビュー中の評価対象となる。ファイルの名前が正しいことを確認すること。



この課題の要件で求められているファイルのみを提出すること。