



C Piscine

C 13

*Summary:* このドキュメントはC Piscine @ 42の C 13モジュール用の課題です。

# 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 : btree_create_node</b>	<b>5</b>
<b>IV</b>	<b>Exercise 01 : btree_apply_prefix</b>	<b>6</b>
<b>V</b>	<b>Exercise 02 : btree_apply_infix</b>	<b>7</b>
<b>VI</b>	<b>Exercise 03 : btree_apply_suffix</b>	<b>8</b>
<b>VII</b>	<b>Exercise 04 : btree_insert_data</b>	<b>9</b>
<b>VIII</b>	<b>Exercise 05 : btree_search_item</b>	<b>10</b>
<b>IX</b>	<b>Exercise 06 : btree_level_count</b>	<b>11</b>
<b>X</b>	<b>Exercise 07 : btree_apply_by_level</b>	<b>12</b>

# 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はこれらのフラッグを用いて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。
- 今後の課題は以下の構造体で取り組んでください。

```
typedef struct      s_btree
{
    struct s_btree  *left;
    struct s_btree  *right;
    void            *item;
}                  t_btree;
```

- ft\_btree.hファイルの中にこの構造体を含めたまま各課題を提出しましょう。
- exercise 01より先は btree\_create\_nodeを使用しますので、あらかじめ準備しておいてください。(プロトタイプをft\_btree.hファイルに入れておくと便利です)。

# Chapter II

## Foreword


Here's the list of releases for **Venom** :

- In League with Satan (single, 1980)
- Welcome to Hell (1981)
- Black Metal (1982)
- Bloodlust (single, 1983)
- Die Hard (single, 1983)
- Warhead (single, 1984)
- At War with Satan (1984)
- Hell at Hammersmith (EP, 1985)
- American Assault (EP, 1985)
- Canadian Assault (EP, 1985)
- French Assault (EP, 1985)
- Japanese Assault (EP, 1985)
- Scandinavian Assault (EP, 1985)
- Manitou (single, 1985)
- Nightmare (single, 1985)
- Possessed (1985)
- German Assault (EP, 1987)
- Calm Before the Storm (1987)
- Prime Evil (1989)
- Tear Your Soul Apart (EP, 1990)
- Temples of Ice (1991)
- The Waste Lands (1992)
- Venom ' 96 (EP, 1996)
- Cast in Stone (1997)
- Resurrection (2000)
- Anti Christ (single, 2006)
- Metal Black (2006)
- Hell (2008)
- Fallen Angels (2011)

Today's subject will seem easier if you listen to **Venom**.

# Chapter III

## Exercise 00 : btree\_create\_node


	Exercise 00
	btree_create_node
	提出するディレクトリ : <i>ex00/</i>
	提出するファイル : <i>btree_create_node.c, ft_btree.h</i>
	使用可能な関数 : <i>malloc</i>

- 新たな要素を作成するために動的にメモリを確保する**btree\_create\_node**関数を作成しましょう。要素の**item**を引数の値に、そしてその他のすべてを0に初期化してください。
- 作成したノードのアドレスが返されます。
- プロトタイプ例

```
t_btree *btree_create_node(void *item);
```

## Chapter IV

### Exercise 01 : btree\_apply\_prefix


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

- 引数として与えられた関数をprefix traversalを利用してツリーを検索し、各ノードのitemに適用させるbtree\_apply\_prefix関数を作成しましょう。
- プロトタイプ例

```
void btree_apply_prefix(t_btree *root, void (*applyf)(void *));
```

# Chapter V

## Exercise 02 : btree\_apply\_infix

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


- 引数として与えられた関数をinfix traversalを利用してツリーを検索し、各ノードのitemに適用させるbtree\_apply\_prefix関数を作成しましょう。
- プロトタイプ例

```
void btree_apply_infix(t_btree *root, void (*applyf)(void *));
```



# Chapter VI

## Exercise 03 : btree\_apply\_suffix


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

- 引数として与えられた関数をsuffix traversalを利用してツリーを検索し、各ノードのitemに適用させるbtree\_apply\_prefix関数を作成しましょう。
- プロトタイプ例

```
void btree_apply_suffix(t_btree *root, void (*applyf)(void *));
```

# Chapter VII

## Exercise 04 : btree\_insert\_data


	Exercise 04
	btree_insert_data
	提出するディレクトリ : <i>ex04/</i>
	提出するファイル : <i>btree_insert_data.c, ft_btree.h</i>
	使用可能な関数 : <i>btree_create_node</i>

- `item`要素をツリーに挿入する**`btree_insert_data`**関数を作成しましょう。引数として渡されたツリーはすでにソート済みです。各 `node`のすべての低層要素は左に配置され、すべての上位または等しい要素は右側に配置されます。引数として `strcmp`に似た比較関数も渡します。
- `root` パラメータは、ツリーのルートノードを指します。初めて呼び出された場合、`NULL`を指します。
- プロトタイプ例

```
void btree_insert_data(t_btree **root, void *item, int (*cmpf)(void *, void *));
```

# Chapter VIII

## Exercise 05 : btree\_search\_item


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

- 引数として与えられた参照データにマッチする、最初の要素を返す**btree\_search\_item**関数を作成しましょう。ツリーは *infix traversal* を使って模索します。要素が見つからない場合は *NULL* を返します。
- プロトタイプ例

```
void *btree_search_item(t_btree *root, void *data_ref, int (*cmpf)(void *, void *));
```

# Chapter IX

## Exercise 06 : btree\_level\_count


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

- 引数として渡された二分木の一番大きい深さを返す**btree\_level\_count**関数を作成しましょう。
- プロトタイプ例

```
int btree_level_count(t_btree *root);
```

# Chapter X

## Exercise 07 : btree\_apply\_by\_level

	Exercise 07
btree_apply_by_level	
提出するディレクトリ : <i>ex07/</i>	
提出するファイル : <i>btree_apply_by_level.c, ft_btree.h</i>	
使用可能な関数 : <i>malloc, free</i>	

- 引数として渡された関数を、各ノードに適用する**btree\_apply\_by\_level**関数を作成しましょう。ツリーはレベルごとに閲覧しなければいけません。呼び出される関数は3つの引数を取ります。
  - void \*型の最初の引数はノードのアイテムです。
  - int型の2つ目の引数は発見時のレベルです : 0はルート,1は子など
  - int型の3つ目の引数は、レベルの最初のnodeの場合、1をそうでない場合は0を意味します。
- プロトタイプ例

```
void btree_apply_by_level(t_btree *root, void (*applyf)(void *item, int current_level, int is_first_elem))
```