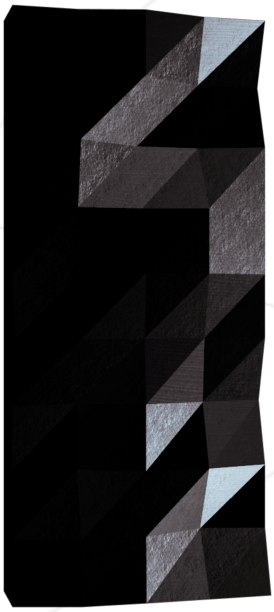

Piscina C

C 13

Resumen: Este documento corresponde a la evaluación del módulo C 13 de la piscina de 42.

Índice general

Capítulo I

Instrucciones

- Esta página será la única referencia: no se fíe de los rumores de pasillo.
- Vuelva a leer bien los enunciados antes de entregar sus ejercicios. Los enunciados pueden cambiar en cualquier momento.
- Tenga cuidado con los permisos de sus archivos y de sus directorios.
- Debe respetar el procedimiento de entrega para todos sus ejercicios.
- Sus compañeros de piscina se encargarán de corregir sus ejercicios.
- Además de por sus compañeros, también será corregido por un programa que se llama la Moulinette.
- La Moulinette es muy estricta a la hora de dar notas. Está completamente automatizada. Es imposible discutir con ella sobre su nota. Por lo tanto, sea extremadamente riguroso para evitar cualquier sorpresa.
- La Moulinette no tiene una mente muy abierta. No intenta comprender el código que no respeta la Norma. La Moulinette utiliza el programa `norminette` para comprobar la norma de sus archivos. Entienda entonces que es estúpido entregar un código que no pase la `norminette`.
- Los ejercicios han sido ordenados con mucha precisión del más sencillo al más complejo. En ningún caso le prestaremos atención ni tendremos en cuenta un ejercicio complejo si no se ha conseguido realizar perfectamente un ejercicio más sencillo.
- El uso de una función prohibida se considera una trampa. Cualquier trampa será sancionada con la nota `-42`.
- Solamente tendrá que entregar una función main() si le pedimos un programa.
- La Moulinette compila con los flags -Wall -Wextra -Werror y utiliza `gcc`.
- Si su programa no compila, tendrá `0`.
- No debe dejar en su directorio ningún archivo que no se haya indicado de forma explícita en los enunciados de los ejercicios.

- ¿Tiene alguna pregunta? Pregunte a su vecino de la derecha. Si no, pruebe con su vecino de la izquierda.
- Su manual de referencia se llama `Google / man / Internet / ...`.
- ¡No olvide participar en el foro Piscina de su Intranet o en el slack de su Piscina!
- Lea detenidamente los ejemplos. Podrían exigir cosas que no se especifican necesariamente en los enunciados...
- Razone. ¡Se lo suplico, por Odín! Maldita sea.
- Para los ejercicios de hoy, utilizaremos la estructura siguiente:

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

- Debe colocar esta estructura en un archivo `ft_btree.h` y entregarlo en cada ejercicio.
- A partir del ejercicio 01 utilizaremos nuestro btree_create_node, tome las medidas necesarias (podría ser interesante tener su prototipo en ft_btree.h...).

Capítulo II

Preámbulo

He aquí la discografía de `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)`

Los enunciados de hoy le resultarán más fáciles si trabaja escuchando `Venom`.

Capítulo III

Ejercicio 00 : btree_create_node

	Ejercicio : 00
btree_create_node	
Directorio de entrega : $ex00/$	
Ficheros a entregar : `btree_create_node.c, ft_btree.h`	
Funciones autorizadas : `malloc`	

- Escriba la función `btree_create_node` que asigne memoria a un elemento nuevo, inicialice su `ítem` con el valor del parámetro y los demás elementos a 0.
- Se retornará la dirección del nodo creado.
- El prototipo de la función deberá ser el siguiente:

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

Capítulo IV

Ejercicio 01 : btree_apply_prefix

	Ejercicio : 01
btree_apply_prefix	
Directorio de entrega : $ex01/$	
Ficheros a entregar : `btree_apply_prefix.c, ft_btree.h`	
Funciones autorizadas : `Ninguna`	

- Escriba la función `btree_apply_prefix` que aplique la función pasada como parámetro al `item` de cada nodo, recorriendo el árbol de manera `prefix`.
- El prototipo de la función deberá ser el siguiente:

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

Capítulo V

Ejercicio 02 : btree_apply_infix

	Ejercicio : 02
btree_apply_infix	
Directorio de entrega : $ex02/$	
Ficheros a entregar : `btree_apply_infix.c, ft_btree.h`	
Funciones autorizadas : `Ninguna`	

- Escriba la función `btree_apply_infix` que aplique la función pasada como parámetro al `item` de cada nodo, recorriendo el árbol de manera `infix`.
- El prototipo de la función deberá ser el siguiente:

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

Capítulo VI

Ejercicio 03 : btree_apply_suffix

	Ejercicio : 03
btree_apply_suffix	
Directorio de entrega : $ex03/$	
Ficheros a entregar : `btree_apply_suffix.c, ft_btree.h`	
Funciones autorizadas : `Ninguna`	

- Escriba la función `btree_apply_suffix` que aplique la función pasada como parámetro al `item` de cada nodo, recorriendo el árbol de manera `suffix`.
- El prototipo de la función deberá ser el siguiente:

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

Capítulo VII

Ejercicio 04 : btree_insert_data

	Ejercicio : 04
btree_insert_data	
Directorio de entrega : $ex04/$	
Ficheros a entregar : `btree_insert_data.c, ft_btree.h`	
Funciones autorizadas : `btree_create_node`	

- Escriba la función `btree_insert_data` que inserte el elemento `item` en un árbol. El árbol pasado como parámetro estará ordenado: en cada `nodo`, se situarán todos los elementos inferiores en la parte izquierda y todos los elementos superiores o iguales en la derecha. Se enviará como parámetro una función de comparación que tenga el mismo comportamiento que `strcmp`.
- El parámetro `root` apunta al nodo de la raíz del árbol. En la primera llamada, apunta a `NULL`.
- El prototipo de la función deberá ser el siguiente:

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

Capítulo VIII

Ejercicio 05 : btree_search_item

	Ejercicio : 05
btree_search_item	
Directorio de entrega : $ex05/$	
Ficheros a entregar : `btree_search_item.c, ft_btree.h`	
Funciones autorizadas : `Ninguna`	

- Escriba la función `btree_search_item` que devuelva el primer elemento correspondiente al dato de referencia pasado como parámetro. Se tendrá que recorrer el árbol de manera `infix`. Si no se encuentra el elemento, la función tendrá que devolver `NULL`.
- El prototipo de la función deberá ser el siguiente:

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

Capítulo IX

Ejercicio 06 : btree_level_count

	Ejercicio : 06
btree_level_count	
Directorio de entrega : $ex06/$	
Ficheros a entregar : `btree_level_count.c, ft_btree.h`	
Funciones autorizadas : `Ninguna`	

- Escriba la función `btree_level_count` que retorne el tamaño de la rama más grande pasada como parámetro.
- El prototipo de la función deberá ser el siguiente:

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

Capítulo X

Ejercicio 07 : btree_apply_by_level

	Ejercicio : 07
btree_apply_by_level	
Directorio de entrega : $ex07/$	
Ficheros a entregar : `btree_apply_by_level.c, ft_btree.h`	
Funciones autorizadas : `malloc, free`	

- Escriba la función `btree_apply_by_level` que aplique la función pasada como parámetro a cada nodo del árbol. Se tendrá que recorrer el árbol nivel a nivel. La función llamada recibirá tres parámetros:
  - El primer parámetro, de tipo `void *`, corresponde al ítem del nodo;
  - El segundo parámetro, de tipo `int`, corresponde al nivel sobre el que nos encontramos: 0 para el root, 1 para los hijos, 2 para los nietos, etc.;
  - El tercer parámetro, de tipo `int`, vale 1 si se trata del primer `nodo` del nivel, si no 0.
- El prototipo de la función deberá ser el siguiente:

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