

Piscina C C 12

Resumen: Este documento corresponde a la evaluación del módulo C 12 de la piscina de 42.

Índice general

1.	Preambulo	
II.	Instrucciones	4
III.	Ejercicio 00 : ft_create_elem	6
IV.	Ejercicio 01 : ft_list_push_front	7
v.	Ejercicio 02 : ft_list_size	8
VI.	Ejercicio 03 : ft_list_last	9
VII.	Ejercicio 04 : ft_list_push_back	10
VIII.	Ejercicio 05 : ft_list_push_strs	11
IX.	Ejercicio 06 : ft_list_clear	12
X.	Ejercicio 07 : ft_list_at	13
XI.	Ejercicio 08 : ft_list_reverse	14
XII.	Ejercicio 09 : ft_list_foreach	15
XIII.	Ejercicio 10 : ft_list_foreach_if	16
XIV.	Ejercicio 11 : ft_list_find	17
XV.	Ejercicio 12 : ft_list_remove_if	18
XVI.	Ejercicio 13 : ft_list_merge	19
XVII.	Ejercicio 14 : ft_list_sort	20
XVIII.	Ejercicio 15 : ft_list_reverse_fun	21
XIX.	Ejercicio 16 : ft_sorted_list_insert	22
XX.	Ejercicio 17 : ft sorted list merge	23

Capítulo I Preámbulo

AVISO DE SPOILER NO LEA LA PÁGINA SIGUIENTE

Usted se lo ha buscado.

- En Star Wars, Dark Vador es el padre de Luke Skywalker.
- En Sospechosos habituales, Verbal es Keyser Soze.
- En El club de la lucha, Tyler Durden y el narrador son la misma persona.
- En El sexto sentido, Bruce Willis está muerto desde el principio.
- En Los otros, los residentes de la casa son los fantasmas y viceversa.
- En Bambi, muere la madre de Bambi.
- En El bosque, los monstruos son los aldeanos y, en realidad, la historia ocurre en nuestra época.
- En Harry Potter, muere Dumbledore.
- En El planeta de los simios, la historia se desarrolla en la Tierra.
- En Juego de tronos, Robb Stark y Joffrey Baratheon mueren en su noche de bodas.
- En Crepúsculo, los vampiros brillan cuando se exponen al sol.
- En Stargate SG-1, Temporada 1, Episodio 18, O'Neill y Carter están en la Antártida.
- En El caballero oscuro: la leyenda renace, Miranda Tate es Talia al Ghul.
- En Super Mario Bros, la princesa se encuentra en otro castillo.

Capítulo II

Instrucciones

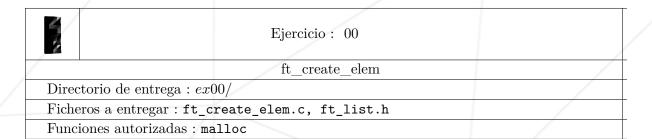
- Esta página será la única referencia: no se fíe de los rumores de pasillo.
- Vuelva a leer bien los enunciados antes de entregar sus ejercicios. Los enunciados pueden cambiar en cualquier momento.
- Tenga cuidado con los permisos de sus archivos y de sus directorios.
- Debe respetar el procedimiento 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.
- \bullet Los ejercicios han sido ordenados con mucha precisión del más sencillo al más complejo. En ning \tilde{A}^{o} n 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.
- <u>No debe</u> dejar en su directorio <u>ningún</u> 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 sobre las listas, utilizaremos la estructura siguiente:

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

Capítulo III

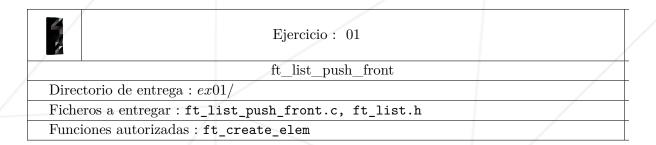
Ejercicio 00 : ft_create_elem



- Escriba la función ft_create_elem que cree un elemento nuevo de tipo t_list.
- Tendrá que asignar data al parámetro proporcionado y next a NULL.
- El prototipo de la función deberá ser el siguiente:

Capítulo IV

Ejercicio 01 : ft_list_push_front

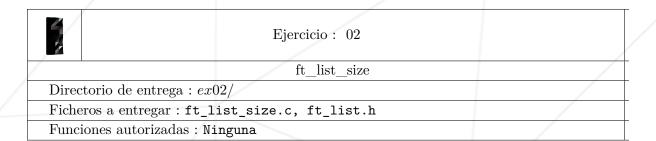


- Escriba la función ft_list_push_front que añada al principio de la lista un elemento nuevo de tipo t_list.
- Tendrá que asignar data al parámetro proporcionado.
- Actualizará, si es preciso, el puntero al principio de la lista.
- El prototipo de la función deberá ser el siguiente:

void ft_list_push_front(t_list **begin_list, void *data);

Capítulo V

Ejercicio 02 : ft_list_size

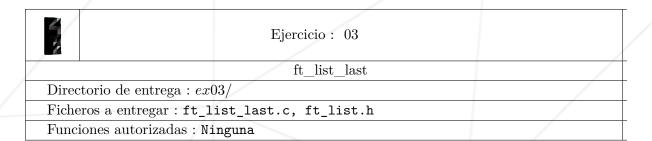


- \bullet Escriba la función ft_list_size que devuelva el n
úmero de elementos de la lista.
- El prototipo de la función deberá ser el siguiente:

int ft_list_size(t_list *begin_list);

Capítulo VI

Ejercicio 03: ft_list_last

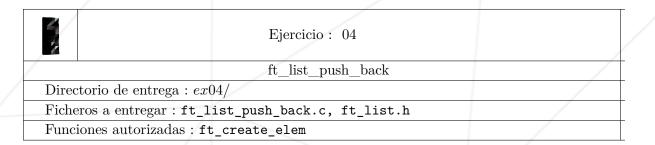


- \bullet Escriba la función ft_list_last que devuelva el \tilde{A}^{o} ltimo elemento de la lista.
- El prototipo de la función deberá ser el siguiente:

t_list *ft_list_last(t_list *begin_list);

Capítulo VII

Ejercicio 04 : ft_list_push_back

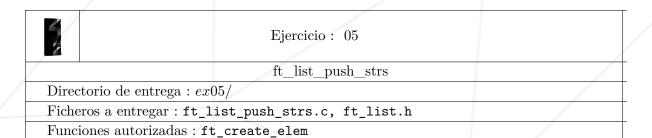


- Escriba la función ft_list_push_back que añada al final de la lista un elemento nuevo de tipo t_list.
- Tendrá que asignar data al parámetro proporcionado.
- Actualizará, si es preciso, el puntero al principio de la lista.
- El prototipo de la función deberá ser el siguiente:

void ft_list_push_back(t_list **begin_list, void *data);

Capítulo VIII

Ejercicio 05 : ft_list_push_strs

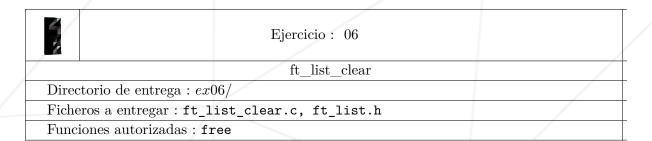


- Escriba la función ft_list_push_strs que cree una lista nueva e introduzca en ella las cadenas de caracteres apuntadas por los elementos de la tabla strs.
- size es el tamaño de strs
- El primer elemento de la tabla se encontrará al final de la lista.
- Se devolverá la dirección del primer elemento de la lista.
- El prototipo de la función deberá ser el siguiente:

t_list *ft_list_push_strs(int size, char **strs);

Capítulo IX

Ejercicio 06: ft_list_clear

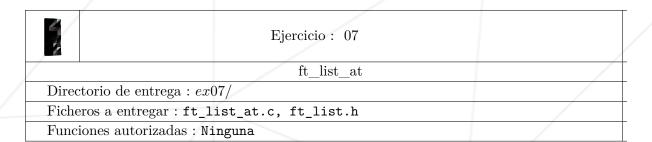


- Escriba la función ft_list_clear que retire y libere todos los elementos de la lista.
- También tendrán que liberarse todos los data usando free_fct.
- El prototipo de la función deberá ser el siguiente:

void ft_list_clear(t_list *begin_list, void (*free_fct)(void *));

Capítulo X

Ejercicio 07: ft_list_at

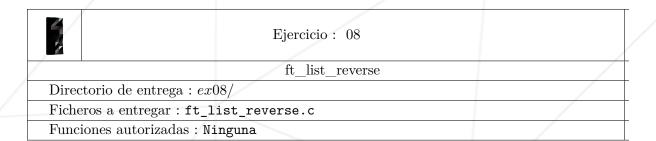


- Escriba la función ft_list_at que devuelva el n-ésimo elemento de la lista, sabiendo que el primer elemento es el elemento 0.
- En caso de error, devolverá un puntero nulo.
- El prototipo de la función deberá ser el siguiente:

t_list *ft_list_at(t_list *begin_list, unsigned int nbr);

Capítulo XI

Ejercicio 08 : ft_list_reverse

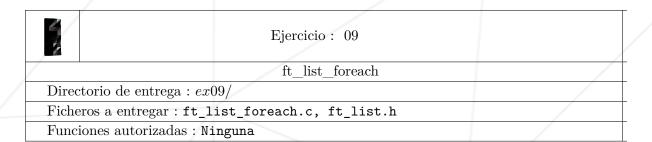


- Escriba la función ft_list_reverse que invierta el orden de los elementos de la lista. Solo se admiten los juegos de punteros.
- Atención, en este ejercicio utilizaremos nuestro propio ft_list.h.
- El prototipo de la función deberá ser el siguiente:

void ft_list_reverse(t_list **begin_list);

Capítulo XII

Ejercicio 09: ft_list_foreach



- Escriba la función ft_list_foreach que aplique una función pasada como parámetro al valor incluido en cada elemento de la lista.
- Se debe aplicar f en el orden de los elementos de la lista.
- El prototipo de la función deberá ser el siguiente:

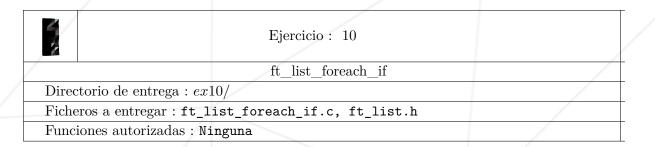
```
void ft_list_foreach(t_list *begin_list, void (*f)(void *));
```

• La función apuntada por f será utilizada de la siguiente forma:

(*f)(list_ptr->data);

Capítulo XIII

Ejercicio 10: ft_list_foreach_if



- Escriba la función ft_list_foreach_if que aplique una función pasada como parámetro al valor incluido en algunos elementos de la lista.
- Solo se aplicará f a los elementos que, al ser pasados como argumento a cmp con data_ref, hagan que cmp devuelva 0.
- Se debe aplicar f en el orden de los elementos de la lista.
- El prototipo de la función deberá ser el siguiente:

```
void ft_list_foreach_if(t_list *begin_list, void (*f)(void *), void
*data_ref, int (*cmp)())
```

 \bullet Las funciones apuntadas por fy por ${\tt cmp}$ serán utilizadas de la siguiente forma:

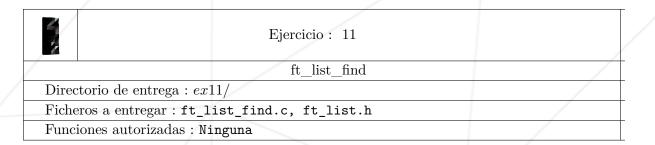
```
(*f)(list_ptr->data);
(*cmp)(list_ptr->data, data_ref);
```



La función cmp podría ser, por ejemplo, ft_strcmp...

Capítulo XIV

Ejercicio 11: ft_list_find



- Escriba la función ft_list_find que devuelva la dirección del primer elemento cuyos datos, cuando se los compare con data_ref usando cmp, hagan que cmp devuelva 0.
- El prototipo de la función deberá ser el siguiente:

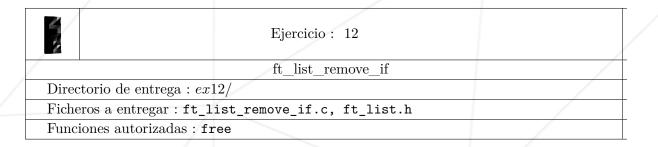
```
t_list *ft_list_find(t_list *begin_list, void *data_ref, int (*cmp)());
```

• La función apuntada por cmp será utilizada de la siguiente forma:

(*cmp)(list_ptr->data, data_ref);

Capítulo XV

Ejercicio 12: ft_list_remove_if



- Escriba la función ft_list_remove_if que borre de la lista todos los elementos cuyos datos, cuando se los compare con data_ref usando cmp, hagan que cmp devuelva 0.
- También tendrán que liberarse todos los data de un elemento que se tenga que borrar usando free_fct.
- El prototipo de la función deberá ser el siguiente:

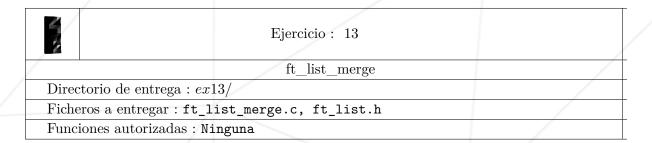
```
void ft_list_remove_if(t_list **begin_list, void *data_ref, int (*cmp)(), void (*free_fct)(void *)
```

• Las funciones apuntadas por free_fct y por cmp serán utilizadas de la siguiente forma:

```
(*cmp)(list_ptr->data, data_ref);
(*free_fct)(list_ptr->data);
```

Capítulo XVI

Ejercicio 13: ft_list_merge

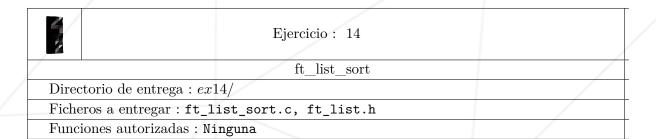


- Escriba la función ft_list_merge que coloque los elementos de una lista begin2 al final de otra lista begin1.
- No se permite la creación de elementos.
- El prototipo de la función deberá ser el siguiente:

void ft_list_merge(t_list **begin_list1, t_list *begin_list2);

Capítulo XVII

Ejercicio 14: ft_list_sort



- Escriba la función ft_list_sort que ordene de forma creciente el contenido de la lista, comparando dos elementos mediante una función de comparación de datos de dos elementos.
- El prototipo de la función deberá ser el siguiente:

```
void ft_list_sort(t_list **begin_list, int (*cmp)());
```

• La función apuntada por cmp será utilizada de la siguiente forma:

```
(*cmp)(list_ptr->data, other_list_ptr->data);
```



La función cmp podría ser, por ejemplo, ft_strcmp.

Capítulo XVIII

Ejercicio 15 : ft__list__reverse__fun

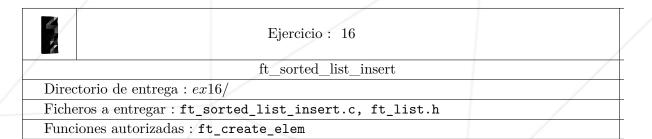
2	Ejercicio: 15	
	ft_list_reverse_fun	
Directorio de entrega : $ex15/$		/
Ficheros a entregar : ft_list_reverse_fun.c, ft_list.h		/
Funciones autorizadas : Ninguna		/

- Escriba la función ft_list_reverse_fun que invierta el orden de los elementos de la lista.
- El prototipo de la función deberá ser el siguiente:

void ft_list_reverse_fun(t_list *begin_list);

Capítulo XIX

Ejercicio 16: ft_sorted_list_insert



- Escriba la función ft_sorted_list_insert que cree un elemento nuevo y lo inserte en una lista ordenada de tal modo que la lista quede en orden creciente.
- El prototipo de la función deberá ser el siguiente:

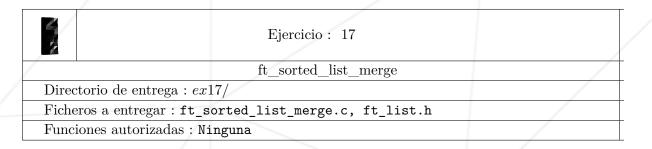
```
void ft_sorted_list_insert(t_list **begin_list, void *data, int (*cmp)());
```

• La función apuntada por cmp será utilizada de la siguiente forma:

(*cmp)(list_ptr->data, other_list_ptr->data);

Capítulo XX

Ejercicio 17: ft_sorted_list_merge



- Escriba la función ft_sorted_list_merge que integre los elementos de una lista ordenada begin2 dentro de otra lista ordenada begin1, de tal modo que la lista begin1 quede en orden creciente.
- El prototipo de la función deberá ser el siguiente:

```
void ft_sorted_list_merge(t_list **begin_list1, t_list *begin_list2, int (*cmp)());
```

• La función apuntada por cmp será utilizada de la siguiente forma:

(*cmp)(list_ptr->data, other_list_ptr->data);