



C Piscine

C 07

Resumen: Este documento corresponde al enunciado del módulo C 07 de la C Piscine de 42.

Versión: 3

Índice general

I.	Instrucciones	2
II.	Introducción	4
III.	Ejercicio 00 : ft_strdup	5
IV.	Ejercicio 01 : ft_range	6
V.	Ejercicio 02 : ft_ultimate_range	7
VI.	Ejercicio 03 : ft_strjoin	8
VII.	Ejercicio 04 : ft_convert_base	9
VIII.	Ejercicio 05 : ft_split	10

Capítulo I

Instrucciones

- Esta página será la única referencia: no te fíes de los rumores.
- ¡Ten cuidado! Los enunciados pueden cambiar en cualquier momento.
- Asegúrate de que tus directorios y archivos tienen los permisos adecuados.
- Debes respetar el procedimiento de entrega para todos tus ejercicios.
- Tus compañeros de piscina se encargarán de corregir tus ejercicios.
- Además de por tus compañeros, también serán corregidos por un programa que se llama la Moulinette.
- La Moulinette es muy estricta a la hora de evaluar. Está completamente automatizada. Es imposible discutir con ella sobre tu nota. Por lo tanto, se 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 en sus archivos. Entiende 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 se tendrá 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 hay que entregar una función `main()` si lo que se pide es un programa.
- La Moulinette compila con los flags `-Wall -Wextra -Werror` y utiliza `gcc`.
- Si tu programa no compila, tendrán un 0.
- No puedes dejar en tu directorio ningún archivo que no se haya indicado de forma explícita en los enunciados de los ejercicios.
- ¿Tienes alguna pregunta? Pregunta a tu compañero de la derecha. Si no, prueba con tu compañero de la izquierda.

- Tu manual de referencia se llama `Google / man / Internet /`
- ¡No olvides participar en el slack de tu Piscina!
- Lee detenidamente los ejemplos. Podrían exigir cosas que no se especifican necesariamente en los enunciados...
- Razona. ¡Te lo suplico, por Thor, por Odín! Maldita sea.



Para este módulo, la Norminette debe ser ejecutada con el flag `-R CheckForbiddenSourceHeader`. La Moulinette también lo utilizará.

Capítulo II


Introducción

He aquí una lista de los monstruos que se pueden encontrar en las famosas mazmorras de Naheulbeuk:

- Todo tipo de zombis;
- Arañas gigantes;
- Orcos;
- Goblins;
- Trolls de las cavernas;
- Brujos;
- Guerreros malditos;
- Ratas mutantes;
- Una botella de aceite;
- Papel higiénico;
- Dos esponjas;
- Raviolis.

Capítulo III

Ejercicio 00 : ft_strdup


	Ejercicio: 00
	ft_strdup
	Directorio de entrega: <i>ex00/</i>
	Archivos a entregar: ft_strdup.c
	Funciones autorizadas: malloc

- Reproduce el comportamiento de la función `strdup` (man `strdup`).
- El prototipo de la función deberá ser el siguiente:

```
char *ft_strdup(char *src);
```

Capítulo IV

Ejercicio 01 : ft_range

	Ejercicio: 01
	ft_range
	Directorio de entrega: <i>ex01/</i>
	Archivos a entregar: ft_range.c
	Funciones autorizadas: malloc


- Escribe una función **ft_range** que devuelva un array de **int**. Este array de **int** contendrá todos los valores entre **min** y **max**.
- **Min** incluido - **max** excluido.
- El prototipo de la función deberá ser el siguiente:

```
int *ft_range(int min, int max);
```

- Si el valor **min** es superior o igual al valor **max**, se devolverá un puntero nulo.

Capítulo V

Ejercicio 02 : ft_ultimate_range

	Ejercicio: 02
	ft_ultimate_range
	Directorio de entrega: <i>ex02/</i>
	Archivos a entregar: ft_ultimate_range.c
	Funciones autorizadas: malloc


- Escribe una función `ft_ultimate_range` que asigne memoria y datos a un array de `int`. Este array de `int` contendrá todos los valores entre `min` y `max`.
- `Min` incluido - `max` excluido.
- El prototipo de la función deberá ser el siguiente:

```
int ft_ultimate_range(int **range, int min, int max);
```

- Devolverá el tamaño de `range` (o -1 en caso de problemas).
- Si el valor `min` es superior o igual al valor `max`, `range` apuntará a `NULL` y retornará 0.

Capítulo VI

Ejercicio 03 : ft_strjoin


	Ejercicio: 03
	ft_strjoin
	Directorio de entrega: <i>ex03/</i>
	Archivos a entregar: ft_strjoin.c
	Funciones autorizadas: malloc

- Escribe una función que concatene el conjunto de cadenas de caracteres apuntadas por **strs**, separándolas por **sep**.
- **size** representa el tamaño de **strs**.
- Si **size** vale 0, habrá que retornar una cadena de caracteres vacía liberable.
- El prototipo de la función deberá ser el siguiente:

```
char *ft_strjoin(int size, char **strs, char *sep);
```

Capítulo VII

Ejercicio 04 : ft_convert_base


	Ejercicio: 04
	ft_convert_base
	Directorio de entrega: <i>ex04/</i>
	Archivos a entregar: ft_convert_base.c , ft_convert_base2.c
	Funciones autorizadas: malloc , free

- Escribe una función que devuelva el resultado de la conversión de la cadena **nbr**, expresada en una base **base_from**, a una base **base_to**.
- **nbr**, **base_from**, **base_to** no son necesariamente modificables.
- **nbr** seguirá las mismas reglas que **ft_atoi_base**. Por lo tanto, cuidado con el + , - y con los espacios blancos.
- El número representado por **nbr** cabe en un **int**.
- Si una base es incorrecta, la función retornará **NULL**.
- Si es preciso que el número devuelto lleve un prefijo, este será un solo y único - , sin espacios ni + .
- El prototipo de la función deberá ser el siguiente:

```
char *ft_convert_base(char *nbr, char *base_from, char *base_to);
```

Capítulo VIII

Ejercicio 05 : ft_split

	Ejercicio: 05
	ft_split
	Directorio de entrega: <i>ex05/</i>
	Archivos a entregar: ft_split.c
	Funciones autorizadas: malloc

- Escribe una función que divida una cadena de caracteres en función de otra cadena de caracteres.
- Habrá que utilizar cada carácter de la cadena `charset` como separador.
- La función retorna un array donde cada uno de sus elementos contiene la dirección de una cadena de caracteres comprendida entre dos separadores. El último elemento del array tendrá que ser igual a 0 para indicar el final del array.
- Tu array no puede tener cadenas vacías. Saca las conclusiones pertinentes.
- La cadena pasada como argumento no será modificable.
- El prototipo de la función deberá ser el siguiente:

```
char **ft_split(char *str, char *charset);
```