



Piscine

07

Staff 42 piscine@42.fr

Résumé: Ce document est le sujet du module C 07 de la piscine C de 42.

Table des matières

I	Consignes	2
II	Pré mbule	4
III	Exercice 00 : ft_strdup	5
IV	Exercice 01 : ft_r nge	6
V	Exercice 02 : ft_ultim te_r nge	7
VI	Exercice 03 : ft_strjoin	8
VII	Exercice 04 : ft_convert_b se	9
VIII	Exercice 05 : ft_split	10

Chapitre I

Consignes

Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.

Relisez bien le sujet avant de rendre vos exercices. tout moment le sujet peut changer.

Attention aux droits de vos fichiers et de vos répertoires.

Vous devez suivre la procédure de rendu pour tous vos exercices.

Vos exercices seront corrigés par vos camarades de piscine.

En plus de vos camarades, vous serez corrigés par un programme appelé la Moulinette.

La Moulinette est très stricte dans sa notation. Elle est totalement automatisée. Il est impossible de discuter de sa note avec elle. Soyez d'une rigueur irréprochable pour éviter les surprises.

La Moulinette n'est pas très ouverte d'esprit. Elle ne cherche pas à comprendre le code qui ne respecte pas la Norme. La Moulinette utilise le programme **norminette** pour vérifier la norme de vos fichiers. Comprenez par là qu'il est stupide de rendre un code qui ne passe pas la **norminette**.

Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.

L'utilisation d'une fonction interdite est un cas de triche. Toute triche est sanctionnée par la note de -42.

Vous ne devrez rendre une fonction `main()` que si nous vous demandons un programme.

La Moulinette compile avec les flags `-Wall -Wextra -Werror`, et utilise `gcc`.

Si votre programme ne compile pas, vous aurez 0.

Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement spécifiés par les énoncés des exercices.

Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec

votre voisin de gauche.

Votre manuel de référence s'appelle **Google / m n / Internet /**

Pensez à discuter sur le forum Piscine de votre Intra, ainsi que sur le slack de votre Piscine!

Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...

Réfléchissez. Par pitié, par Odin! Nom d'une pipe.



Pour cette journée, la norminette doit être lancée avec le flag `-R`
`CheckForbiddenSourceHeader`. La moulinette l'utiliser aussi.

Chapitre II


Préambule

Voici une liste des monstres que l'on peut trouver dans le célèbre Donjon de Naheulbeuk :

- Toutes sortes de morts-vivants ;
- Des rognées géantes ;
- Des orques ;
- Des gobelins ;
- Des trolls dans les souterrains ;
- Des sorciers ;
- Des guerriers maudits ;
- Des rats mutants ;
- Une bouteille d'huile ;
- Du papier toilette ;
- Deux éponges ;
- Des rats violis.

Chapitre III

Exercice 00 : ft_strdup

	Exercice : 00
ft_strdup	
Dossier de rendu : ex00	
Fichiers à rendre : ft_strdup.c	
Fonctions autorisées : malloc	


Reproduire à l'identique le fonctionnement de la fonction `strdup` (man strdup).

Elle devra être prototypée de la façon suivante :

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

Chapitre IV

Exercice 01 : ft_range

	Exercice : 01
	ft_range
	Dossier de rendu : <i>ex01</i>
	Fichiers à rendre : ft_range.c
	Fonctions autorisées : malloc

Écrire une fonction **ft_range** qui retourne un tableau d'**int**. Ce tableau d'**int** contiendra toutes les valeurs entre **min** et **max**.

min inclu - **max** exclu.


Elle devra être prototypée de la façon suivante :

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

Si la valeur **min** est supérieure ou égale à la valeur **max**, un pointeur nul sera retourné.

Chapitre V

Exercice 02 : ft_ultimate_range

	Exercice : 02
	ft_ultimate_range
	Dossier de rendu : ex02
	Fichiers à rendre : ft_ultimate_range.c
	Fonctions autorisées : malloc

Écrire une fonction `ft_ultimate_range` qui alloue et assigne un tableau d'`int`. Ce tableau d'`int` contiendra toutes les valeurs entre `min` et `max`.

`min` inclus - `max` exclu.

Elle devra être prototypée de la façon suivante :


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

La taille de `range` sera retournée (ou -1 en cas de problème).

Si la valeur `min` est supérieure ou égale à la valeur `max`, `range` pointerà sur NULL et on renverra 0.

Chapitre VI

Exercice 03 : ft_strjoin

	Exercice : 03
	ft_strjoin
	Dossier de rendu : <i>ex03</i>
	Fichiers à rendre : ft_strjoin.c
	Fonctions autorisées : malloc

Écrire une fonction qui va concatener l'ensemble des chaînes de caractères pointées par **strs** en les séparants à l'aide de **sep**.

size représente la taille de **strs**.


Si **size** vaut 0, il faut retourner une chaîne de caractères vide que l'on peut `free()`.

Elle devra être prototypée de la façon suivante :

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

Chapitre VII

Exercice 04 : ft_convert_base

	Exercice : 04
	ft_convert_base
	Dossier de rendu : ex04
	Fichiers à rendre : ft_convert_base.c, ft_convert_base2.c
	Fonctions autorisées : malloc, free

Écrire une fonction qui renvoie le résultat de la conversion de la chaîne `nbr` exprimée en une base `b_se_from` dans une base `b_se_to`.

`nbr`, `b_se_from`, `b_se_to` ne seront pas forcément modifiable.

`nbr` suivra les mêmes règles que `ft_atoi_base`. Attention donc au '+', '-' et aux whitespaces.

Le nombre représenté par `nbr` tient dans un `int`.


Si une base est incorrecte, la fonction renverra `NULL`.

Le nombre retourné doit être préfixé seulement par un seul et unique '-' si c'est nécessaire, pas de whitespaces ou de '+'.
Elle devra être prototypée de la façon suivante :

```
char *ft_convert_base(char *nbr, char *b_se_from, char *b_se_to);
```

Chapitre VIII

Exercice 05 : ft_split

	Exercice : 05
	ft_split
	Dossier de rendu : <i>ex05</i>
	Fichiers à rendre : ft_split.c
	Fonctions autorisées : malloc

Écrire une fonction qui découpe une chaîne de caractères en fonction d'une autre chaîne de caractères.

Il faudra utiliser chaque caractère de la chaîne **charset** comme séparateur.

La fonction renvoie un tableau où chaque élément de celui-ci contient l'adresse d'une chaîne de caractères comprise entre deux séparateurs. Le dernier élément du tableau devra être égal à 0 pour marquer la fin du tableau.

Il ne doit pas y avoir de chaîne vide dans votre tableau. Tirez-en les conclusions qui s'imposent.

La chaîne qui sera transmise ne sera pas modifiable.

Elle devra être prototypée de la façon suivante :

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