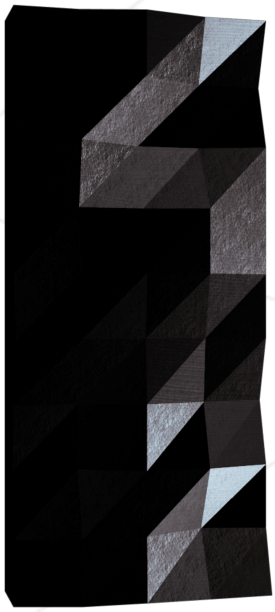

Piscine C

Jour 07

Staff 42 piscine@42.fr

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

Table des matières

Chapitre I

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- 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. Comprendre 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`.
- Si ft_putchar() est une fonction autorisée, nous compilerons avec notre `ft_putchar.c`.
- 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 specifiés 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 / man / Internet / ...`.
- Pensez à discuter sur le forum Piscine de votre Intra !
- 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.

`Attention aux droits !`

`Pour cette journée, la norminette doit être lancée avec le flag` $--CheckForbiddenSourceHeader$`. La moulinette l'utilisera 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 araignées 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 raviolis.

Chapitre III

Exercice 00 : ft_strdup

	Exercice : 00
ft_strdup	
Dossier de rendu : $ex00/$	
Fichiers à rendre : `ft_strdup.c`	
Fonctions Autorisées : `malloc`	
Remarques : `n/a`	

- 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 V

Exercice 02 : ft_ultimate_range

	Exercice : 02
ft_ultimate_range	
Dossier de rendu : *ex*02/	
Fichiers à rendre : `ft_ultimate_range.c`	
Fonctions Autorisées : `malloc`	
Remarques : `n/a`	

- 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` inclu - `max` exclu.
- Elle devra être prototypée de la façon suivante :

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

- Si la valeur `min` est supérieure ou égale à la valeur `max`, `range` pointera sur NULL.
- La taille de `range` sera retournée (ou 0 en cas de problème).

Chapitre VI

Exercice 03 : ft_concat_params

	Exercice : 03
ft_concat_params	
Dossier de rendu : $ex03/$	
Fichiers à rendre : `ft_concat_params.c`	
Fonctions Autorisées : `malloc`	
Remarques : `n/a`	

- Écrire une fonction qui tranforme les arguments reçus en ligne de commande en une unique chaîne de caractères. Les arguments seront separés par un "\n".
- Elle devra être prototypée de la façon suivante :

```
char *ft_concat_params(int argc, char **argv);
```

Chapitre VII

Exercice 04 : ft_split_whitespaces

	Exercice : 04
ft_split_whitespaces	
Dossier de rendu : $ex04/$	
Fichiers à rendre : `ft_split_whitespaces.c`	
Fonctions Autorisées : `malloc`	
Remarques : `n/a`	

- Écrire une fonction qui découpe une chaîne de caractères en mots.
- Les séparateurs sont les espaces, les tabulations et les retours à la ligne.
- La fonction renvoie un tableau où chaque case contient l'adresse d'une chaîne de caractères représentant un mot. Le dernier élement 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_whitespaces(char *str);
```

Chapitre VIII

Exercice 05 : ft_print_words_tables

	Exercice : 05
ft_print_words_tables	
Dossier de rendu : $ex05/$	
Fichiers à rendre : `ft_print_words_tables.c`	
Fonctions Autorisées : `ft_putchar`	
Remarques : `n/a`	

- Écrire une fonction qui affiche le contenu du tableau créé par la fonction de l'exercice précédent.
- Chaque mot sera seul sur une ligne.
- Chaque mot sera suivi d'un "\n", y compris le dernier.
- Cet exercice sera compilé avec votre ft_split_whitespaces.c
- Faites attention à ne pas faire de `multiple define`.
- Elle devra être prototypée de la façon suivante :

```
void ft_print_words_tables(char **tab);
```

Chapitre IX

Exercice 06 : ft_convert_base

	Exercice : 06
ft_convert_base	
Dossier de rendu : $ex06/$	
Fichiers à rendre : `ft_convert_base.c`	
Fonctions Autorisées : `malloc, free`	
Remarques : `n/a`	

- Écrire une fonction qui renvoie le résultat de la conversion de la chaîne `nbr` exprimée en une base `base_from` dans une base `base_to` sous forme d'une chaîne de caractères allouée avec suffisamment de mémoire. Le nombre representé par `nbr` tient dans un `int`.
- Elle devra être prototypée de la façon suivante :

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

Chapitre X

Exercice 07 : ft_split

	Exercice : 07
ft_split	
Dossier de rendu : $ex07/$	
Fichiers à rendre : `ft_split.c`	
Fonctions Autorisées : `malloc`	
Remarques : `n/a`	

- Écrire une fonction qui découpe une chaîne de caractères en fonction d'une autre chaîne de caractères.
- Il faudra utiliser chaque caractère de la chaine `charset` comme séparateur.
- La fonction renvoie un tableau où chaque case contient l'adresse d'une chaîne de caractères comprise entre deux séparateur. Le dernier élement 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);
```