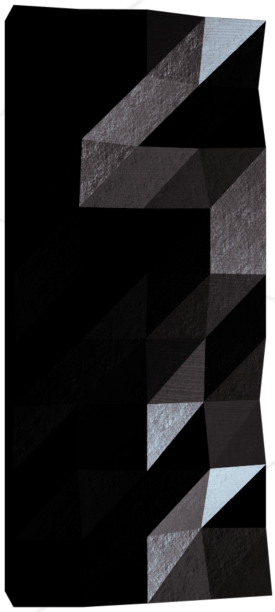

Piscine C

Jour 03

Staff 42 piscine@42.fr

Résumé: Ce document est le sujet du jour 03 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.

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

Chapitre II

Préambule

Le jeu du Sirop selon `Perceval`, tiré de la série *Kaamelott* :

"Bon, j'vais vous apprendre les règles simplifiées, parce que les vraies règles, elles sont velues. Bon, le seul truc, c'est que normalement, ça se joue à trois. Mais c'est pas grave on va se débrouiller.

Le principe, c'est de faire des valeurs. Donc là, mettons, on est trois, il y a trois valeurs à distribuer. On va dire, sirop de huit, sirop de quatorze et sirop de vingt-et-un. Vous occupez pas des sirops tout de suite. Ce qu'il faut comprendre d'abord, c'est les valeurs. Si vous lancez une valeur en début de tour, mettons un sirop de huit, pour commencer petit, les autres ont le choix entre laisser filer la mise ou relancer un sirop de quatorze. On tourne dans le sens des valeurs. C'est pour ça, il faut bien comprendre le système des valeurs; après, ça va tout seul.

Bon alors mettons que j'ouvre avec un sirop de huit.
Si c'est vous qu'avez siroté au tour d'avant, ça tourne dans votre sens. Alors soit vous laissez filer, vous dites "file-sirop", soit vous vous sentez de relancer et vous annoncez un sirop de quatorze. Comme on a commencé les annonces, le second joueur a pas le droit de laisser filer. Vous pouvez soit relancer un sirop de vingt-et-un, soit vous abandonnez le tour et vous dites "couche-sirop" ou "sirop Jeannot", ça dépend des régions. Et après, soit on fait la partie soit je fais un "contre-sirop" ! Et à partir de là, sirop de pomme sur vingt-et-un donc on fait la partie en quatre tours jusqu'à qu'il y en ait un qui sirote.

À la gagne, il n'y a que trois possibilités : soit vous faites votre sirop de huit, vous dites "beau sirop" et on recompte, soit vous faites votre sirop de quatorze, vous dites "beau sirop, sirop gagnant" et on vous rajoute la moitié, soit vous faites votre sirop de vingt-et-un et vous dites "beau sirop, mi-sirop, siroté, gagne-sirop, sirop-grelot, passe-montagne, sirop au bon goût".

Normalement ça se joue avec des cartes mais si vous avez que des dés, vous pouvez aussi jouer avec des dés puisque ce qui compte c'est les valeurs."

Au moins un des exercices suivants n'a aucun rapport avec le jeu du Sirop.

Chapitre III

Exercice 00 : ft_ft

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

- Écrire une fonction qui prend un pointeur sur int en paramètre et donne à l'int la valeur de 42.
- Elle devra être prototypée de la façon suivante :

```
void        ft_ft(int *nbr);
```

Chapitre IV

Exercice 01 : ft_ultimate_ft

	Exercice : 01
ft_ultimate_ft	
Dossier de rendu : *ex*01/	
Fichiers à rendre : `ft_ultimate_ft.c`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Écrire une fonction qui prend un pointeur sur pointeur sur pointeur sur pointeur sur pointeur sur pointeur sur pointeur sur pointeur sur pointeur sur int en paramètre et donne à l'int la valeur de 42.
- Elle devra être prototypée de la façon suivante :

```
void        ft_ultimate_ft(int *********nbr);
```

Chapitre V

Exercice 02 : ft_swap

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

- Écrire une fonction qui échange le contenu de deux entiers dont les adresses sont données en paramètres.
- Elle devra être prototypée de la façon suivante :

```
void    ft_swap(int *a, int *b);
```

Chapitre VI

Exercice 03 : ft_div_mod

	Exercice : 03
ft_div_mod	
Dossier de rendu : *ex*03/	
Fichiers à rendre : `ft_div_mod.c`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Écrire une fonction `ft_div_mod` qui a le prototypage suivant :

```
void    ft_div_mod(int a, int b, int *div, int *mod);
```

- Cette fonction divise les deux paramètres `a` et `b` et stocke le resultat dans l'int pointé par `div`.
  Elle stocke également le reste de la division de `a` et `b` dans l'int pointé par `mod`.

Chapitre VII

Exercice 04 : ft_ultimate_div_mod

	Exercice : 04
ft_ultimate_div_mod	
Dossier de rendu : *ex*04/	
Fichiers à rendre : `ft_ultimate_div_mod.c`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Écrire une fonction `ft_ultimate_div_mod` qui a le prototypage suivant :

```
void    ft_ultimate_div_mod(int *a, int *b);
```

- Cette fonction divise les int pointés par `a` et `b`.
  Le résultat de la division est stocké dans l'int pointé par `a`.
  Le résultat du reste de la division est stocké dans l'int pointé par `b`.

Chapitre VIII

Exercice 05 : ft_putstr

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

- Écrire une fonction qui affiche un à un les caractères d'une chaîne à l'écran.
- L'adresse du premier caractère de la chaîne est contenue dans le pointeur passé en paramètre à la fonction.
- Elle devra être prototypée de la façon suivante :

```
void    ft_putstr(char *str);
```

Chapitre IX

Exercice 06 : ft_strlen

	Exercice : 06
ft_strlen	
Dossier de rendu : $ex06/$	
Fichiers à rendre : `ft_strlen.c`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Écrire une fonction qui compte le nombre de caractères dans une chaîne de caractères et qui retourne le nombre trouvé.
- Elle devra être prototypée de la façon suivante :

```
int         ft_strlen(char *str);
```

Chapitre X

Exercice 07 : ft_strrev

	Exercice : 07
ft_strrev	
Dossier de rendu : *ex*07/	
Fichiers à rendre : `ft_strrev.c`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Écrire une fonction qui inverse une chaîne de caractères.
- Elle devra renvoyer str.
- Elle devra être prototypée de la façon suivante :

```
char        *ft_strrev(char *str);
```

- Exemple :

```
a => a
ab => ba
abcde => edcba
```

Chapitre XI

Exercice 08 : ft_atoi

	Exercice : 08
ft_atoi	
Dossier de rendu : *ex*08/	
Fichiers à rendre : `ft_atoi.c`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Reproduire à l'identique le fonctionnement de la fonction `atoi` (man atoi).
- Elle devra être prototypée de la façon suivante :

```
int     ft_atoi(char *str);
```

Chapitre XII

Exercice 09 : ft_sort_integer_table

	Exercice : 09
ft_sort_integer_table	
Dossier de rendu : $ex09/$	
Fichiers à rendre : `ft_sort_integer_table.c`	
Fonctions Autorisées : `Aucune`	
Remarques : `n/a`	

- Écrire une fonction qui trie un tableau d'entiers par ordre croissant.
- Les paramètres sont un pointeur sur entier et le nombre d'entiers dans le tableau.
- La fonction devra être prototypée de la façon suivante :

```
void    ft_sort_integer_table(int *tab, int size);
```