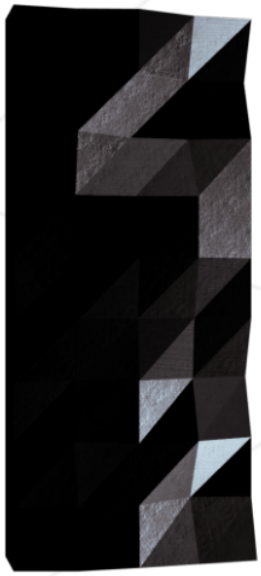

Piscine

04

Résumé: ce document est le sujet du module C 04 de l piscine C de 42.

T ble des m tières

Ch pitre 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.
- ttention 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`.
- 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, 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 CheckForbiddenSourceHe der. La moulinette l'utilisera aussi.`

Ch pitre II

Pré mbule

Voici les paroles du générique de `Nicky Larson` :

```
Une ombre file dans la nuit
C'est un assassin qui s'enfuit
Et comme un démon il sourit
Son crime restera impuni
Une voiture qui surgit
Un coup de frein, des pneus qui crient
Un coup de feu qui retentit
La justice s'appelle Nicky

[Refrain]
Dans la chaleur
De la nuit
Le mal est toujours puni
 ucun danger ne l'impressionne
Les coups durs il les affectionne
Et la justice le passionne
Nicky Larson ne craint personne
Lorsque les coups de feu résonnent
Comme un eclair il tourbillone
Surtout si la fille est mignonne
Nicky Larson ne craint personne

Comme un chasseur il suit sa proie
Pour que la justice et le droit
Triomphent, il est prêt à donner
Toute sa vie sans hésiter
Quand sa silhouette apparaît
Les méchants se mettent à trembler
Ils savent qu'ils ne pourront jamais
Echapper à ce justicier

[Refrain]
```

Ce sujet n'a, malheureusement, rien à voir avec `Nicky Larson`.

Ch pitre III

Exercice 00 : ft_strlen

	Exercice : 00
ft_strlen	
Dossier de rendu : $ex00$	
Fichiers à rendre : `ft_strlen.c`	
Fonctions utorisées : `ucune`	

- 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(ch r *str);
```

Ch pitre IV

Exercice 01 : ft_putstr

	Exercice : 01
ft_putstr	
Dossier de rendu : $ex01$	
Fichiers à rendre : `ft_putstr.c`	
Fonctions utorisées : `write`	

- 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(ch r *str);
```

Ch pitre V

Exercice 02 : ft_putnbr

	Exercice : 02
ft_putnbr	
Dossier de rendu : $ex02$	
Fichiers à rendre : `ft_putnbr.c`	
Fonctions utorisées : `write`	

- Écrire une fonction qui affiche un nombre passé en paramètre. La fonction devra être capable d'afficher la totalité des valeurs possibles dans une variable de type `int`.
- Elle devra être prototypée de la façon suivante :

```
void ft_putnbr(int nb);
```

- Par exemple :
  `ft_putnbr(42)` affiche "42".

Ch pitre VI

Exercice 03 : ft_ toi

	Exercice : 03
ft_atoi	
Dossier de rendu : $ex03$	
Fichiers à rendre : `ft_atoi.c`	
Fonctions utorisées : `ucune`	

- Ecrire une fonction qui convertit le début de la chaîne pointée par str en entier de type int
- str peut commencer par un nombre arbitraire de 'white space' (comme defini par `isspace(3)`)
- str peut ensuite être suivi par un nombre arbitraire de signe + et de signe -. Le signe - fera changer le signe de l'entier retourné en fonction du nombre de signe - et si celui ci est pair ou impair.
- Pour finir str devra être composée de chiffre de la base 10
- Votre fonction devra lire str tant que celle ci suit les règles au dessus et elle doit retourner le nombre trouvé jusque là.
- Vous ne devriez pas prendre en compte les overflows et les underflows, le résultat est considérer comme indéfini dans ces cas.
- Vous pouvez comparer votre fonction avec la vrai fonction atoi à part la partie sur les signes ainsi que l'overflow.
- Voici l'exemple d'un programme qui affiche la valeur de retour de atoi :

```
$>./ .out "   ---+--+1234 b567"
-1234
```

- Elle devra être prototypée de la façon suivante :

```
int     ft_ toi(ch r *str);
```

Ch pitre VII

Exercice 04 : ft_putnbr_b se

	Exercice : 04
ft_putnbr_base	
Dossier de rendu : $ex04$	
Fichiers à rendre : `ft_putnbr_base.c`	
Fonctions utorisées : `write`	

- Écrire une fonction qui affiche un nombre dans le terminal dans une base donnée.
- Ce nombre est fourni sous la forme d'un `int` et la base sous la forme d'une `chaîne de caractères`.
- La base contient tous les symboles utilisables pour afficher le nombre :
  `0123456789` est la base couramment utilisée pour représenter nos nombres décimaux ;

  `01` est une base binaire ;

  `0123456789 BCDEF` est une base hexadecimale ;

  `poneyvif` est une base octale.
- La fonction doit gérer les nombres négatifs.
- Si un paramètre contient une erreur la fonction n'affiche rien. Une erreur peut être :
  base est vide ou est de taille 1 ;

  base contient deux fois le même caractère ;

  base contient les caractères + ou -.
- Elle devra être prototypée de la façon suivante :

```
void        ft_putnbr_b se(int nbr, ch r *b se);
```

Ch pitre VIII

Exercice 05 : ft_ toi_b se

	Exercice : 05
ft_atoi_base	
Dossier de rendu : $ex05$	
Fichiers à rendre : `ft_atoi_base.c`	
Fonctions utorisées : `ucune`	

- Ecrire une fonction qui convertit le début de la chaîne pointée par str en entier de type int.
- str est dans une base specifique passée en second argument de la fonction.
- part le système de base, cette fonction doit reproduire le comportement de ft_atoi.
- Si un paramètre contient une erreur la fonction renvoie 0. Une erreur peut être :
  la base est vide ou est de taille 1 ;

  la base contient deux fois le même caractère ;

  la base contient les caractères + ou - ou des whitespaces ;
- Elle devra être prototypée de la façon suivante :

```
int       ft_ toi_b se(ch r *str, ch r *b se);
```