



Sujet de l'examen 03

Version 1.0

Bocal bocal@42.fr
Marvin marvin@42.fr

Résumé: Ce document est le sujet du quatrième examen.

Table des matières

I	Détails administratifs	2
I.1	Consignes générales	2
I.2	Le Code	3
II	Exercices	4
II.1	Exercice 00 - aff_z	4
II.2	Exercice 01 - ft_print_numbers	5
II.3	Exercice 02 - ft_putstr	6
II.4	Exercice 03 - ft_strlen	7
II.5	Exercice 04 - ft_swap	8
II.6	Exercice 05 - ft_djamal	9
II.7	Exercice 06 - ft_strcapitalize	10
II.8	Exercice 07 - aff_first_param	11
II.9	Exercice 08 - ulstr	12
II.10	Exercice 09 - ft_strcpy	13
II.11	Exercice 10 - half_str	14
II.12	Exercice 11 - rotone	15
II.13	Exercice 12 - ft_strcmp	16
II.14	Exercice 13 - wdmatch	17
II.15	Exercice 14 - ft_strrev	18
II.16	Exercice 15 - union	19
II.17	Exercice 16 - ft_strdup	20
II.18	Exercice 17 - ft_range	21
II.19	Exercice 18 - inter	22
II.20	Exercice 19 - epur_str	23
II.21	Exercice 20 - rostring	24
II.22	Exercice 21 - tab_mult	25
II.23	Exercice 22 - str_maxlenoc	26
II.24	Exercice 23 - ft_sort_params	27
II.25	Exercice 24 - ft_split	28
II.26	Exercice 25 - ft_list_size	29
II.27	Exercice 26 - ft_list_foreach	30
II.28	Exercice 27 - count_island	31
II.29	Exercice 28 - Secu1	35
II.30	Exercice 29 - Secu2	36
II.31	Exercice 30 - Secu3	37

Chapitre I

Détails administratifs

I.1 Consignes générales

- Aucune forme de communication n'est permise.
- Ceci est un examen, il est interdit de discuter, écouter de la musique, faire du bruit ou produire toute autre nuisance pouvant déranger les autres étudiants ou perturber le bon déroulement de l'examen.
- Vos téléphones portables et autres appareils technologiques doivent être éteints et rangés hors d'atteinte. Si un téléphone sonne, toute la rangée concernée est éliminée et doit sortir immédiatement.
- Votre home contient deux dossiers : "rendu" et "sujet".
- Le répertoire "sujet" contient le sujet de l'examen.
- Le répertoire "rendu" est un clone de votre dépôt de rendu dédié à cet examen. Vous y ferez vos commits et vos pushes.
- Seul le contenu que vous avez pushé sur votre dépôt de rendu sera corrigé.
- Vous ne pouvez exécuter les programmes que vous avez compilés vous-même que dans votre dossier "rendu" et ses sous-dossiers. Cela est interdit ailleurs.
- Chaque exercice doit être réalisé dans le repertoire correspondant au nom indiqué dans l'en-tête de chaque exercice.
- Vous devez rendre, à la racine du repertoire "rendu", un fichier nommé "auteur" comprenant votre login suivi d'un retour à la ligne. Si ce fichier est absent ou mal formaté, vous ne serez pas corrigé. Par exemple :

```
$> cat -e ~/rendu/auteur
xlogin$
$>
```

- Certaines notions nécessaires à la réalisation de certains exercices sont à découvrir dans les mans.
- C'est un programme qui s'occupe du ramassage, respectez les noms, les chemins, les fichiers et les répertoires...

- Tout fichier en trop dans un exercice entrainera un 0 à celui ci.
- En cas de problème technique avec le sujet, on ne doit s'adresser qu'au surveillant uniquement. Interdiction de parler à ses voisins.
- En cas de question, on ne doit s'adresser qu'au surveillant uniquement. Interdiction de parler à ses voisins.
- Tout matériel non explicitement autorisé est implicitement interdit.
- Exceptionnnellement, les exercices de cet examen peuvent être réalisés dans l'ordre que vous souhaitez. De plus, la correction ne s'arrêtera pas au premier exercice faux.
- Toute sortie de la salle est définitive.


I.2 Le Code

- Des fonctions utiles ou des fichiers supplémentaires sont parfois donnés dans des sous répertoires de ~/sujet/. Si ce dossier n'existe pas ou bien s'il est vide, c'est que nous ne vous fournissons rien.
- La correction du code est automatisée. Un programme testera le bon fonctionnement des exercices : la "Moulinette".
- Les fonctions autorisées sont indiquées dans l'en-tête de chaque exercice. Vous pouvez recoder toutes les fonctions qui vous semblent utiles à votre guise.
- Toute fonction non autorisée explicitement est implicitement interdite.
- Vous avez le droit à des feuilles blanches et un stylo. Pas de cahier de notes, de pense-bête ou autres cours. Vous êtes seuls face à votre examen.
- Pour toute question après l'examen, créez un ticket sur le dashboard (dashboard.42.fr).

Chapitre II

Exercices

II.1 Exercice 00 - aff_z


	Exercice : 00
	aff_z
	Dossier de rendu : <i>ex00/</i>
	Fichiers à rendre : aff_z.c
	Fonctions Autorisées : write
	Remarques : n/a

Écrire une fonction nommée "aff_z" qui prend en paramètre une chaîne de caractères et qui affiche sur la sortie standard le premier caractère 'z' rencontré suivi d'un retour à la ligne. Si aucun 'z' n'est rencontré dans la chaîne, la fonction affiche 'z' suivi d'un retour à la ligne.

La fonction sera prototypée de la manière suivante :

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


II.2 Exercice 01 - ft_print_numbers

	Exercice : 01
	ft_print_numbers
	Dossier de rendu : <i>ex01/</i>
	Fichiers à rendre : ft_print_numbers.c
	Fonctions Autorisées : write
	Remarques : n/a

- Écrire une fonction qui affiche tous les chiffres dans l'ordre croissant.
- Elle devra être prototypée de la façon suivante :

```
void ft_print_numbers(void);
```

II.3 Exercice 02 - ft_putstr

	Exercice : 02
	ft_putstr
	Dossier de rendu : <i>ex02/</i>
	Fichiers à rendre : ft_putstr.c
	Fonctions Autorisées : write
	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);
```


II.4 Exercice 03 - ft_strlen

	Exercice : 03
	ft_strlen
	Dossier de rendu : <i>ex03/</i>
	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);
```


II.5 Exercice 04 - ft_swap


	Exercice : 04
	ft_swap
	Dossier de rendu : <i>ex04/</i>
	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);
```


II.6 Exercice 05 - ft_djamal

	Exercice : 05
	ft_djamal
	Dossier de rendu : <i>ex05/</i>
	Fichiers à rendre : ft_djamal.c
	Fonctions Autorisées : Aucune
	Remarques : n/a

- Ecrire une fonction `ft_djamal` qui prendra en paramètre trois `int` et retournera la valeur médiane.
- Cette fonction sera prototypée de la façon suivante :

```
int ft_djamal(int i, int j, int k);
```

II.7 Exercice 06 - ft_strcapitalize

	Exercice : 06
	ft_strcapitalize
	Dossier de rendu : <i>ex06/</i>
	Fichiers à rendre : ft_strcapitalize.c
	Fonctions Autorisées : write
	Remarques : n/a

Écrire un programme qui met en majuscule la première lettre de chaque mot et le reste du mot en minuscule.


Un mot est une suite de caractères alphanumériques.

```
$>./ft_strcapitalize "salut, comment tu vas ? 42mots quarante-deux; cinquante+et+un" | cat -e
Salut, Comment Tu Vas ? 42mots Quarante-Deux; Cinquante+Et+Un$
$>

$>./ft_strcapitalize "La verite est la seule source de courage."| cat -e
La Verite Est La Seule Source De Courage.$
$>

$>./ft_strcapitalize | cat -e
$
$>
```

II.8 Exercice 07 - aff_first_param

	Exercice : 07
	aff_first_param
	Dossier de rendu : <i>ex07/</i>
	Fichiers à rendre : aff_first_param.c
	Fonctions Autorisées : write
	Remarques : n/a


Vous devez écrire un programme qui affiche uniquement le premier argument qui lui est fourni suivi d'un retour à la ligne. Si aucun argument n'est fourni, vous devez afficher uniquement un retour à la ligne.

Exemples :

```
$>./aff_first_param bonjour les users
bonjour
$>./aff_first_param

$>./aff_first_param 5 4 3 2 1 | cat -e
5$
$>./aff_first_param | cat -e
$
$>
```

II.9 Exercice 08 - ulstr

	Exercice : 08
ulstr	
Dossier de rendu : <i>ex08/</i>	
Fichiers à rendre : ulstr.c	
Fonctions Autorisées : write	
Remarques : n/a	

Écrire un programme qui prend en paramètre une chaîne de caractères, qui transforme toutes les minuscules en majuscules et toutes les majuscules en minuscules. Les autres caractères restent inchangés.


Ce programme doit afficher le résultat sur la sortie standard suivi d'un retour à la ligne.

Le programme doit renvoyer un retour à la ligne si il n'y a aucun paramètre.

Exemples :

```
$>./ulstr "L'eSPrit nE peUt plUs pRogResSer s'Il staGne et sI peRsIsTent VAnIte et auto-
justification." | cat -e
l'EsPRIT Ne PEuT PLuS PRoGrESSER S'iL STAGNE ET Si PERSiStENT vaNiTE ET AUTO-JUSTIFICATION.$
$>./ulstr "S'enTOuRer dE sECreT eSt uN sIGnE De mAnQuE De coNNaiSSanCe. " | cat -e
s'ENToUrER De SecREt Est Un SigNe dE MaNqUe dE COnnAIssANcE. $
$>./ulstr | cat -e
$
$>
```


II.10 Exercice 09 - ft_strcpy

	Exercice : 09
	ft_strcpy
	Dossier de rendu : <i>ex09/</i>
	Fichiers à rendre : ft_strcpy.c
	Fonctions Autorisées : Aucune
	Remarques : n/a

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

```
char *ft_strcpy(char *s1, char *s2);
```

II.11 Exercice 10 - half_str

	Exercice : 10
	half_str
	Dossier de rendu : <i>ex10/</i>
	Fichiers à rendre : half_str.c
	Fonctions Autorisées : write
	Remarques : n/a

Écrire un programme qui prend en paramètre une ou plusieurs chaines de caractères et qui affiche chaque chaine dans l'ordre un caractère sur deux.

Ce programme doit afficher le résultat sur la sortie standard suivi d'un retour à la ligne.

L'affichage se termine toujours par un retour à la ligne.


Le programme doit renvoyer un retour à la ligne si il n'y a aucun paramètre.

Exemples :

```
$>./half_str "abc"
ac
$>
$>./half_str "test de phrase" "deuxieme test" "un avant dernier test" "" "enfin c'est finis" |
cat -e
ts eprs$
duim et$
u vn ene et$
$
efncetfns$
$>
$>./half_str "a btcfdceqf gehdiajikhllmгнаovpyqer srttuuvawcxayaz" "142637415869728395" "
" "Baodnxnce cshja4nycrew qaq dtdoduqsq" | cat -e
abcdefghijklmnopqrstuvwxyz$
123456789$
$
Bonne chance a tous$
$>

$>./half_str | cat -e
$
$>
```

II.12 Exercice 11 - rotone

	Exercice : 11
rotone	
Dossier de rendu : <i>ex11/</i>	
Fichiers à rendre : rotone.c	
Fonctions Autorisées : write	
Remarques : n/a	

Écrire un programme nommé "rotone" qui prend en paramètre une chaîne de caractères et qui affiche cette chaîne en remplaçant chaque caractère alphabétique par le caractère suivant dans l'ordre alphabétique.

'z' devient 'a' et 'Z' devient 'A'. Les majuscules restent des majuscules et les minuscules restent des minuscules.


L'affichage se termine toujours par un retour à la ligne.

Si aucun paramètre n'est transmis, le programme affiche un retour à la ligne.

Exemple :

```
$>./rotone "abc"
bcd
$>./rotone "Les stagiaires du staff ne sentent pas toujours tres bon." | cat -e
Mft tubhjbjsft ev tubgg of tfoufou qbt upvkvst usft cpo.$
$>./rotone "AkjhZ zLKIJz , 23y " | cat -e
BlkiA aMLJKa , 23z $
$>./rotone | cat -e
$
$>
```



II.13 Exercice 12 - ft_strcmp

	Exercice : 12
	ft_strcmp
	Dossier de rendu : <i>ex12/</i>
	Fichiers à rendre : ft_strcmp.c
	Fonctions Autorisées : Aucune
	Remarques : n/a

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

```
int      ft_strcmp(char *s1, char *s2);
```

II.14 Exercice 13 - wdmatch

	Exercice : 13
wdmatch	
Dossier de rendu : <i>ex13/</i>	
Fichiers à rendre : wdmatch.c	
Fonctions Autorisées : write	
Remarques : n/a	

Le programme prend en paramètres deux chaînes de caractères et vérifie qu'il est possible d'écrire la première chaîne de caractères à l'aide des caractères de la deuxième chaîne, tout en respectant l'ordre des caractères dans la deuxième chaîne.


Si cela est possible, le programme renvoie la première chaîne de caractères suivi d'un retour à la ligne.

Le programme renvoie un retour à la ligne s'il n'y a aucun paramètre ou si leur nombre est différent de deux.

Exemple :

```
$>./wdmatch "faya" "fgvvfdxcacpolhyghbreda" | cat -e
faya$
$>./wdmatch "quarante deux" "qfqfsudf arzgsayns tsregfdgs sjytdekuoixq " | cat -e
quarante deux$
$>./wdmatch "error" rrerrrrfiiljdxfxyuifrrvcoojh | cat -e
$
$>./wdmatch | cat -e
$
$>
```

II.15 Exercice 14 - ft_strrev

	Exercice : 14
	ft_strrev
	Dossier de rendu : <i>ex14/</i>
	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
```

II.16 Exercice 15 - union

	Exercice : 15
union	
Dossier de rendu : <i>ex15/</i>	
Fichiers à rendre : union.c	
Fonctions Autorisées : write	
Remarques : n/a	

Écrire un programme qui prend en paramètre deux chaînes de caractères et qui affiche sans doublon les caractères qui apparaissent dans l'une ou dans l'autre.

L'affichage se fera dans l'ordre d'apparition dans la ligne de commande.


L'affichage doit être suivi d'un retour à la ligne.

S'il n'y a pas deux paramètres le programme affiche un retour à la ligne.

Exemple :

```
$>./union zpadinton "paqefwtdjetyiytjneytjoeyjnejejj" | cat -e
zpadintoqefwjy$
$>./union ddf6vewg64f gtwthgdwthdwfteewhrtag6h4ffdhsd | cat -e
ddf6vewg4thras$
$>./union "rien" "cette phrase ne cache rien" | cat -e
rienct phas$
$>./union | cat -e
$
$>
```


II.17 Exercice 16 - ft_strdup

	Exercice : 16
	ft_strdup
	Dossier de rendu : <i>ex16/</i>
	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 *s);
```

II.18 Exercice 17 - ft_range

	Exercice : 17
	ft_range
	Dossier de rendu : <i>ex17/</i>
	Fichiers à rendre : ft_range.c
	Fonctions Autorisées : malloc
	Remarques : n/a


- 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.

```
Pour ft_range(3, 10) le resultat sera un tableau avec [3, 4, 5, 6, 7, 8, 9]  
Pour ft_range(0, 1) le resultat sera un tableau avec [0]
```


II.19 Exercice 18 - inter

	Exercice : 18
inter	
Dossier de rendu : <i>ex18/</i>	
Fichiers à rendre : inter.c	
Fonctions Autorisées : malloc, write, free	
Remarques : n/a	

- Écrire un programme qui prend en parametre deux chaines de caracteres et qui affiche sans doublon les caracteres communs au deux chaines.
- L’affichage se fera dans l’ordre d’apparition dans la premiere chaine. L’affichage doit etre suivi d’un retour à la ligne.
- Si il n y a pas deux parametres le programme affiche d’un retour à la ligne.
- Exemple

```
$>./inter padinton "paqefwtdjetyiytjneytjoeyjnejejj" | cat -e
padinto$
$>
$>./inter ddf6vewg64f gtwthgdwthdwfteewhrtag6h4ffdhsd | cat -e
df6ewg4$
$>
$>./inter "rien" "cette phrase ne cache rien" | cat -e
rien$
$>
$>./inter | cat -e
$
$>
```


II.20 Exercice 19 - epur_str

	Exercice : 19
epur_str	
Dossier de rendu : <i>ex19/</i>	
Fichiers à rendre : epur_str.c	
Fonctions Autorisées : malloc, write, free	
Remarques : n/a	

- Écrire un programme qui prend en paramètre une chaîne de caractères et qui affiche les mots séparés par un seul espace.
- Le dernier mot sera suivi d'un retour à la ligne (même s'il y en a aucun).
- Il ne devra y avoir d'espace ni avant le premier n'y après le dernier mot.
- On appelle "mot" une chaîne de caractères séparée par soit des espaces et tabulations, soit le début de la chaîne ou la fin de la chaîne.
- Si aucun paramètre n'est transmis, `epur_str` affiche un retour à la ligne.

```
$>./epur_str "abc cba abc cab cba" | cat -e
abc cba abc cab cba$
$>
$>./epur_str "  Remus  et  Romulus  sont  les deux mamelles de Rome  " | cat -e
Remus et Romulus sont les deux mamelles de Rome$
$>
$>./epur_str | cat -e
$
$>
```



II.21 Exercice 20 - rostring

	Exercice : 20
rostring	
Dossier de rendu : <i>ex20/</i>	
Fichiers à rendre : rostring.c	
Fonctions Autorisées : write, malloc, free	
Remarques : n/a	

- Écrire un programme qui prend en parametres une chaine de caracteres et qui affiche cette chaine en procedant a une rotation de celle-ci de droite a gauche.
- Ainsi le premier mot se retrouve le dernier et l'ordre des autres n'est pas modifies.
- Les mots sont des chaines de caracteres separees par des espaces et/ou des tabulations.
- Les mots sont affiches separees par un seul et unique espace.
- L'affichage sera suivie d'un retour à la ligne.
- Si aucun paramètre n'est transmis, rostring affiche un retour à la ligne.
- Exemples :

```
$>./rostring "abc  " | cat -e
abc$
$>
$>./rostring "Que la      lumiere soit et la lumiere fut"
la lumiere soit et la lumiere fut Que
$>
$>./rostring "      AkjhZ zLKIJz , 23y"
zLKIJz , 23y AkjhZ
$>
$>./rostring | cat -e
$
$>
```

II.22 Exercice 21 - tab_mult

	Exercice : 21
	tab_mult
	Dossier de rendu : <i>ex21/</i>
	Fichiers à rendre : tab_mult.c
	Fonctions Autorisées : write
	Remarques : n/a

- Il s'agit de réaliser un programme qui affiche la table de multiplication.
- Ce programme prend en parametre une chaine de caractere et affiche le resultat sur la sortie standard.
- La chaine de carateres placée en parametre sera forcement un nombre.
- Le nombre est strictement positif et rentre dans un int
- Le nombre * 9 tiendra dans un int
- Si il n'y a pas de paramètre le programme renvoie un retour à la ligne.
- Exemple 1 :

```
$>./tab_mult 9
1 x 9 = 9
2 x 9 = 18
3 x 9 = 27
4 x 9 = 36
5 x 9 = 45
6 x 9 = 54
7 x 9 = 63
8 x 9 = 72
9 x 9 = 81
$>
```


- Exemple 2 :

```
$>./tab_mult 19
1 x 19 = 19
2 x 19 = 38
3 x 19 = 57
4 x 19 = 76
5 x 19 = 95
6 x 19 = 114
7 x 19 = 133
8 x 19 = 152
9 x 19 = 171
$>
```

- Exemple 3 :

```
$>./tab_mult | cat -e
$
$>
```

II.23 Exercice 22 - str_maxlenoc

	Exercice : 22
	str_maxlenoc
	Dossier de rendu : <i>ex22/</i>
	Fichiers à rendre : str_maxlenoc.c
	Fonctions Autorisées : write, malloc, free
	Remarques : n/a

- Écrire le programme `str_maxlenoc` qui prend en parametres `n` chaines de caracteres et qui affiche, suivi d'un retour à la ligne, la plus grande chaine de caracteres incluse dans toutes les chaines passees en parametres.
- Si plusieurs chaines correspondent, on affichera celle qui apparait en premier dans le premier parametre.
- `"` est forcement dans toutes les chaines.
- Si aucun parametre n'est transmis, `str_maxlenoc` affiche un retour à la ligne.
- On dit que `A` est inclus dans `B` avec `A` et `B` des chaines de caracteres si `A` est une sous-chaine de `B` ou si `A` et `B` sont identiques.
- Exemple :

```
$> ./str_maxlenoc ab bac abacabccabcb
a
$>


$> ./str_maxlenoc bonjour salut bonjour bonjour
u
$>

$> ./str_maxlenoc xoxAoxo xoxAox oxAox oxo A ooxAoxx oxooxo Aox | cat -e
$
$
$>

$> ./str_maxlenoc bosdsdfnjodur atehhellosd afkuonjosurafg headfgllosf fghellosag
afdfbosnjourafg
os
$>

$> ./str_maxlenoc | cat -e
$
$
$>
```

II.24 Exercice 23 - ft_sort_params


	Exercice : 23
	ft_sort_params
	Dossier de rendu : <i>ex23/</i>
	Fichiers à rendre : ft_sort_params.c
	Fonctions Autorisées : write
	Remarques : n/a

Écrire un programme qui affiche les arguments reçus en ligne de commande triés par ordre ascii. Chaque argument devra être sur une ligne séparée. Si aucun argument n'est passé au programme, affichez un retour à la ligne.

Exemples :

```
$>./ft_sort_params ga bu zo | cat -e
bu$
ga$
zo$
$>./ft_sort_params | cat -e
$
$>
```

II.25 Exercice 24 - ft_split

	Exercice : 24
	ft_split
	Dossier de rendu : <i>ex24/</i>
	Fichiers à rendre : ft_split.c
	Fonctions Autorisées : malloc
	Remarques : n/a


Écrire une fonction qui prend en paramètre une chaîne de caractères et qui la découpe en mots. On appelle mots des suites de caractères séparés par des espaces, des tabulations ou un retour à la ligne ou le début ou la fin de la chaîne.

Le tableau sera terminé par un NULL.

Cette fonction devra avoir le prototype suivant :

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

II.26 Exercice 25 - ft_list_size

	Exercice : 25
	ft_list_size
	Dossier de rendu : <i>ex25/</i>
	Fichiers à rendre : <code>ft_list_size.c</code> , <code>ft_list.h</code>
	Fonctions Autorisées : Aucune
	Remarques : n/a


- Pour cet exercice, nous utiliserons la structure suivante :

```
typedef struct    s_list
{
    struct s_list *next;
    void        *data;
}                t_list;
```

- Vous devez mettre cette structure dans le fichier `ft_list.h` à rendre.
- Écrire la fonction `ft_list_size` qui renvoie le nombre d'éléments dans la liste.
- Elle devra être prototypée de la façon suivante :

```
int ft_list_size(t_list *begin_list);
```

II.27 Exercice 26 - ft_list_foreach

	Exercice : 26
	ft_list_foreach
	Dossier de rendu : <i>ex26/</i>
	Fichiers à rendre : <code>ft_list_foreach.c</code> , <code>ft_list.h</code>
	Fonctions Autorisées : Aucune
	Remarques : n/a

- Pour cet exercice, nous utiliserons la structure suivante :

```
typedef struct    s_list
{
    struct s_list *next;
    void        *data;
}                t_list;
```


- Vous devez mettre cette structure dans le fichier `ft_list.h` à rendre.
- Écrire la fonction `ft_list_foreach` qui applique une fonction donnée en paramètre à l'information contenue dans chaque maillon de la liste.
- Elle devra être prototypée de la façon suivante :

```
void    ft_list_foreach(t_list *begin_list, void (*f)(void *));
```

- La fonction pointée par `f` sera utilisée de la façon suivante :

```
(*f)(list_ptr->data);
```

II.28 Exercice 27 - count_island

	Exercice : 27
	count_island
	Dossier de rendu : <i>ex27/</i>
	Fichiers à rendre : *
	Fonctions Autorisées : <code>open</code> , <code>close</code> , <code>read</code> , <code>write</code> , <code>malloc</code> , <code>free</code>
	Remarques : n/a

- Le programme prend en parametre un fichier contenant une serie de lignes de longueurs egales contenant soit le caractere '.' soit le caractere 'X'. Ces lignes forment un rectangle de '.' comportant des ilots de 'X'.
- Une ligne est une suite de caracteres '.' et de caracteres 'X' qui se termine par un retour à la ligne. Les lignes font toutes la meme taille. La taille maximum d'une ligne est 1024 caracteres.
- Une colonne de caracteres est formee par l'ensemble des caracteres dans un fichier qui sont separees par le meme nombre de caracteres du debut de leur ligne respective.
- On dit que deux caracteres se touchent s'ils sont :
 - soit sur la meme ligne et contigus.
 - soit sur la meme colone et sur des lignes contigues.
- Un ilot de 'X' est forme par l'ensemble des caracteres qui se touchent.
- Le programme doit parcourir le fichier ligne par ligne et l'afficher a l'ecran en remplaçant tous les 'X' des ilots par leur numero d'apparition dans le fichier. Le programme devra effectue ce traitement en commancant par le debut du fichier.
- Il ne peut pas y avoir deux resultats differents pour un meme fichier.
- Si le fichier est vide, qu'une erreur s'est produite ou que aucun fichier n'est passe en parametre, le programme ecrit simplement le caractere du retour à la ligne sur sa sortie standard.
- Le fichier comporte au maximum 10 ilots.
- Vous trouverez dans le repertoire misc des exemples de fichier.
- Votre programme sera compilé avec la commande : `gcc -Wall -Wextra -Werror *.c`

- Exemple :


```
$>cat toto
.....XXXXXXXX.....
.....XXXXXXXX.....XXXXXXXX.....
.....XXXXXXXX.....XX.....XXXX
.....XXXXX.....X.....XXXXXXXXXX
.....X.....
.....XXXXXXXXXXXXX.....X.....
.....X.....XXXXXXXXXX
.....X.....XXXXXXXXXXXX
.....X.....
XX.....XXXX
XX.....XXXXXXXXXXXX.....X
.....
.....X
.....XXXX.....XX
$>

$>./count_island toto
.....00000000.....
.....00000000.....11111111.....
.....00000000.....111.....11111
.....000000.....2.....111111111111
.....2.....
.....333333333333.....2.....
.....3.....22222222.....
.....3.....2222222222.....
.....3.....
44.....5555
44.....666666666666.....5
.....
.....7
.....88888.....77
$>
```

```
$>cat qui_est_la
.....
...X.....X....XXXXX.....XXXXXXX..XXXXXXXXXX..XXXXXXXXXX.....
...XX.....XX....XX...XX....XX....XX....XXXX....XXXXXXXXXX.....
...XXXX.XXXX..XX....XX..XX....XX....XX....XX.....
...XX.XXXX.XX...XX....XX..XX....XX....XX....XX.....
...XX..X..XX..XX....XX..XXXXXXXXX.....XX....XXXXX.....
...XX.....XX..XXXXXXXXXX..XXXX....XX....XXXXX.....
...XX.....XX..XX.....XX..XX.XX.....XX....XX.....
...XX.....XX..XX.....XX..XX..X.....XX....XX.....
...XX.....XX..XX.....XX..XX..X....XXXXXX..XXXXXXXXXX.....
...XX.....XX.XX.....XX.XX....XX..XXXXXXXXXX..XXXXXXXXXX..X...
.....
$>
$>./count_island qui_est_la
.....
...0.....0....11111.....2222222..3333333333..4444444444.....
...00.....00....11...11...22....22....3333....4444444444.....
...0000..0000...11....11..22....22....33....44.....
...00.0000.00...11....11..22....22....33....44.....
...00...0..00...11....11..22222222.....33....44444.....
...00.....00...111111111..2222.....33....44444.....
...00.....00..11.....11..22.22.....33....44.....
...00.....00..11.....11..22...5.....33....44.....
...00.....00..11.....11..22...6.....333333..4444444444.....
...00.....00.11.....11.22....77..3333333333..4444444444..8...
.....
$>
```

```
$> ./count_island | cat -e
$
$>
```

II.29 Exercice 28 - Secu1


	Exercice : 28
Secu1	
Dossier de rendu : <i>ex28/</i>	
Fichiers à rendre : secret	
Fonctions Autorisées : Tout	
Remarques : n/a	



Faites attention à ne pas passer trop de temps sur cet exercice.

Vous trouverez une tarball **secu1.tar.gz** dans le répertoire **misc/** de cet examen. Cette tarball contient un ensemble de fichiers, dont un nommé **secret**. Vous devez rendre ce fichier pour valider cet exercice.

II.30 Exercice 29 - Secu2


	Exercice : 29
Secu2	
Dossier de rendu : <i>ex29/</i>	
Fichiers à rendre : secret	
Fonctions Autorisées : Tout	
Remarques : n/a	



Faites attention à ne pas passer trop de temps sur cet exercice.

Vous trouverez un exécutable nommé **secu2** dans le répertoire **misc/** de cet examen. Quand vous exécutez ce binaire, un mot de passe vous est réclamé. Votre travail consiste à trouver ce mot de passe par n'importe quel moyen. Une fois le bon mot de passe entré, le binaire affiche une phrase secrète que vous devrez copier telle quelle dans un fichier nommé **secret** sans aucun caractère supplémentaire. Vous devez rendre ce fichier pour valider cet exercice.

II.31 Exercice 30 - Secu3

	Exercice : 30
Secu3	
Dossier de rendu : <i>ex30/</i>	
Fichiers à rendre : secret	
Fonctions Autorisées : Tout	
Remarques : n/a	



Faites attention à ne pas passer trop de temps sur cet exercice.

Vous trouverez un exécutable nommé **secu3** dans le répertoire **misc/** de cet examen. Quand vous exécutez ce binaire, un mot de passe vous est réclamé. Votre travail consiste à trouver ce mot de passe par n'importe quel moyen. Une fois le bon mot de passe entré, le binaire affiche une phrase secrète que vous devrez copier telle quelle dans un fichier nommé **secret** sans aucun caractère supplémentaire. Vous devez rendre ce fichier pour valider cet exercice.