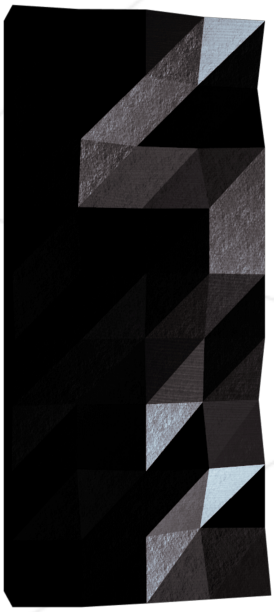

Sujet de l'examen 01

Version 1.0

Bocal bocal@42.fr

Résumé: Ce document est le sujet du second examen du module algo-1-001. Cet examen sera pour vous et pour nous l'occasion de faire un point sur votre avancement.

Table des matières

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. Vous avez du le trouver puisque vous lisez ce document.
- Le répertoire "rendu" est un clone de votre dépot de rendu dédié à cet examen. Vous y ferez vos commits et vos pushs.
- Seul le contenu que vous avez pushé sur votre dépot 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 nécéssaires 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...

- Lorsqu'un exercice impose un nom de fichier, seul ce fichier sera ramassé. Sinon l'exercice stipule `*.c, *.h` et tous vos fichiers `.c` et `.h` seront ramassés, et ces fichiers là seulement.
- 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.
- La correction s'arrêtera au premier exercice faux.
- Toute sortie de la salle est définitive.
- Un surveillant peut vous expulser de la salle s'il le juge nécéssaire.

I.2 Le Code

- Des fonctions utiles ou des fichiers supplémentaires sont parfois donnés dans un sous repertoires de ~/sujet/. Si ce dossier n'existe pas ou bien s'il est vide, c'est que nous ne vous fournissons rien. Ce dossier sera généralement nommé `misc`, mais celà peut varier d'un examen à l'autre.
- La correction du code est automatisée. Un programme testera le bon fonctionnement des exercices : la "Moulinette".
- Lorsqu'un exercice vous demande d'écrire un programme avec un ou plusieurs fichiers nommés, votre programme sera compilé avec la commande `gcc -Wall -Wextra -Werror ficher1.c fichier2.c fichiern.c -o nom_programme`.
- Lorsqu'un exercice vous demande d'écrire un programme et laisse les noms et le nombre de fichiers à votre discrétion, votre programme sera compilé avec la commande : `gcc -Wall -Wextra -Werror *.c -o nom_programme`.
- Enfin, lorqu'un exercice vous demande de rendre une fonction (et donc un seul fichier nommé), votre fichier sera compilé avec la commande `gcc -c -Wall -Wextra -Werror votrefichier.c`, puis nous compilerons notre `main` et linkerons l'éxécutable.
- 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. Utiliser une fonction qui n'est pas autorisée est de la triche sanctionnée par un `-42`.
- 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 - ulstr

	Exercice : 00
ulstr	
Dossier de rendu : *ex*00/	
Fichiers à rendre : `ulstr.c`	
Fonctions Autorisées : `write`	
Remarques : `n/a`	

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

Vous devez afficher le résultat suivi d'un `'\n'`.

Si le nombre de paramètres transmis est différent de 1, le programme affiche `'\n'`.

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.  $
$>./ulstr "3:21 Ba  tOut  moUn ki Ka di KE m'en Ka fe fot" | cat -e
3:21 bA  ToUT  MOuN KI kA DI ke M'EN kA FE FOT$
$>./ulstr | cat -e
$
```

II.2 Exercice 01 - wdmatch

	Exercice : 01
wdmatch	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `wdmatch.c`	
Fonctions Autorisées : `write`	
Remarques : `n/a`	

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

Si cela est possible, le programme affiche la première chaine de caractères suivi de `'\n'`. Sinon le programme affiche seulement `'\n'`.

Si le nombre de paramètres transmis est différent de 2, le programme affiche `'\n'`.

Exemples :

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

II.3 Exercice 02 - ft_strdup

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

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

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

II.4 Exercice 03 - inter

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

Ecrire un programme qui prend en paramètres deux chaines de caractères et qui affiche sans doublon les caractères communs aux deux chaines.

L'affichage se fera dans l'ordre d'apparition dans la premiere chaine. L'affichage doit etre suivi d'un `'\n'`.

Si le nombre de paramètres transmis est différent de 2, le programme affiche `'\n'`.

Exemples :

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

II.5 Exercice 04 - str_maxlenoc

	Exercice : 04
str_maxlenoc	
Dossier de rendu : *ex*04/	
Fichiers à rendre : `str_maxlenoc.c`	
Fonctions Autorisées : `write, malloc, free`	
Remarques : `n/a`	

Ecrire un programme qui prend en parametres n chaines de caractères et qui affiche, suivi d'un retour à la ligne, la plus grande chaine de caractères incluse dans toutes les chaines passées en paramètres. Si plusieurs chaines correspondent, on affichera celle qui apparait en premier dans le premier paramètre. A noter que "" 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.

Exemples :

```
$>./str_maxlenoc ab bac abacabccabcb
a
$>./str_maxlenoc bonjour salut bonjour bonjour
u
$>./str_maxlenoc xoxAoxo xoxAox  oxAox oxo  A ooxAoxx oxooxo Axo | cat -e
$
$>./str_maxlenoc bosdsdfnjodur atehhellosd afkuonjosurafg headfgllosf fghellosag afdfbosnjourafg
os
$>./str_maxlenoc | cat -e
$
```

II.6 Exercice 05 - count_island

	Exercice : 05
count_island	
Dossier de rendu : *ex*05/	
Fichiers à rendre : `*.c, *.h`	
Fonctions Autorisées : `open, close, read, write, malloc, free`	
Remarques : `n/a`	

- Le programme prend en paramètre un fichier contenant une serie de lignes de longueurs égales 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 caractères '.' et de caracteres 'X' qui se termine par un retour à la ligne. Les lignes font toutes la même taille. La taille maximum d'une ligne est 1024 caractères.
- Une colonne de caractères est formée par l'ensemble des caractères dans un fichier qui sont séparés par le même nombre de caractères du début de leur ligne respective.
- On dit que deux caractères se touchent s'ils sont soit sur la même ligne et contigus, soit sur la même colone et sur des lignes contigues.
- Un ilot de 'X' est forme par l'ensemble des caractères qui se touchent.
- Le programme doit parcourir le fichier ligne par ligne et l'afficher à l'écran en remplacant tous les 'X' des ilots par leur numéro d'apparition dans le fichier. Le programme devra effectuer ce traitement en commancant par le debut du fichier.
- Il ne peut pas y avoir deux résultats différents pour un même fichier.
- Si le fichier est vide, qu'une erreur s'est produite (lignes de taille différentes par exemple) ou que aucun fichier n'est passé en paramètre, le programme ecrit simplement le caractère 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.

Exemple :

```
$>cat toto
.................XXXXXXXX.....................................
...................XXXXXXXXX.......XXXXXXXX.....................
.................XXXXXXXX.............XXX...XXXXX.................
.....................XXXXXX.....X...XXXXXXXXXXX...................
.................................X.....................................
......XXXXXXXXXXXX.............X.....................................
..................X............XXXXXXXXX.............................
..................X.........XXXXXXXXXXX.............................
..................X.....................................................
XX.....................................................XXXX
XX..................XXXXXXXXXXXX.............................X
.....................................................................
.....................................................................X.
....................XXXXX.....................................XX
$>
$>./count_island toto
.................00000000.....................................
...................000000000.......11111111.....................
.................00000000.............111...11111.................
.....................000000.....2...11111111111...................
.................................2.....................................
......3333333333333.............2.....................................
..................3............222222222.............................
..................3.........222222222222.............................
..................3.....................................................
44.....................................................5555
44..................6666666666666.............................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...XXXXXXXX.......XX......XXXXX...........
...XX......XX...XXXXXXXXX...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...
.....................................................................
$>
```

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

II.7 Exercice 06 - g_diam

Exercice : 06
g_diam
Dossier de rendu : *ex*06/
Fichiers à rendre : `*.c, *.h`
Fonctions Autorisées : `write, malloc, free`
Remarques : `n/a`

Le programme prend en paramètre une chaine de caractères. Cette chaine représente un graphe et est composée d'une suite d'arêtes entre les noeuds de ce graphe. Les arêtes sont séparées par un espace. Les noeuds sont représentées par des nombres et les arête par deux noeuds séparés par '-'. Par exemple, s'il éxiste une arête entre le noeud 2 et le noeud 3, les representations possibles de cette arête seront soit `"2-3"`, soit `"3-2"`.

Le programme devra afficher le nombre de noeuds du plus long chemin suivit d'un `'\n'` sachant qu'il est impossible de revenir en arrière. c'est-à-dire de passer par un noeud plus d'une fois. Voir les exemples çi-dessous.

Si le nombre de paramètres transmis est différent de 1, le programme affiche `'\n'`.

```
$>./g_diam "17-5 5-8 8-2 2-8 2-8 17-21 21-2 5-2 2-6 6-14 6-12 12-19 19-14 14-42" | cat -e
10$
$>./g_diam "1-2 2-3 4-5 5-6 6-7 7-8 9-13 13-10 10-2 10-11 11-12 12-8 16-4 16-11 21-8 21-12 18-10 18-13
    21-18" | cat -e
15$
```

II.8 Exercice 07 - Time Lord

	Exercice : 07
Time Lord	
Dossier de rendu : $ex07/$	
Fichiers à rendre : `secret`	
Fonctions Autorisées : `Tout, absolument tout ce que vous pouvez imaginer`	
Remarques : `n/a`	

Vous trouverez un executable nommé `time_lord` dans le répertoire `misc/` de cet examen. Quand vous éxécutez ce binaire, il affiche le nombre de secondes restantes avant d'afficher la phrase secrète. Quand le nombre de secondes est dépassé, le binaire affiche la phrase secrète. Votre travail consiste à trouver cette phrase secrète par n'importe quel moyen. Vous devez copier la phrase secrète telle quelle sans AUCUN caractère supplémentaire dans un fichier nommé `secret`. Vous devez rendre ce fichier avec la bonne phrase pour valider cet exercice.