



Sujet de l'examen 00

Version 1.0

Bocal bocal@42.fr

Résumé: Ce document est le sujet du premier 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

| | | |
|-----------|--------------------------------------|----------|
| I | Détails administratifs | 2 |
| I.1 | Consignes générales | 2 |
| I.2 | Le Code | 3 |
| II | Exercices | 5 |
| II.1 | Exercice 00 - rotone | 5 |
| II.2 | Exercice 01 - hiddenp | 6 |
| II.3 | Exercice 02 - union | 7 |
| II.4 | Exercice 03 - ord_alphlong | 8 |
| II.5 | Exercice 04 - g_diam | 9 |
| II.6 | Exercice 05 - count_island | 10 |
| II.7 | Exercice 06 - Time Lord | 12 |

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 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...

- 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 nécessaire.

I.2 Le Code


- Des fonctions utiles ou des fichiers supplémentaires sont parfois donnés dans un sous répertoire 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 cela 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 fichier1.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, lorsqu'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'exécutable.
- 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 - rotone

| | |
|---|---------------|
|  | Exercice : 00 |
| rotone | |
| Dossier de rendu : <i>ex00/</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 le nombre de paramètres transmis est différent de 1, le programme affiche '\n'.

Exemple :

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

II.2 Exercice 01 - hiddenp

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

Écrire un programme nommé "hiddenp" qui prend en paramètres deux chaînes de caractères et qui affiche 1 suivi de '\n' si la chaîne du premier paramètre est cachée dans la deuxième chaîne. Sinon il affiche 0 suivi de '\n'.


Soit s1 et s2 des chaînes de caractères. On dit que la chaîne s1 est cachée dans la chaîne s2 si on peut trouver chaque caractère de s1 dans s2 et ce dans le même ordre que dans s1.

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

Exemples :

```
$>./hiddenp "fgex.;" "tyf34gdgf;'ektufjhgdgex.;.rtjynur6" | cat -e
1$
$>./hiddenp "abc" "2altrb53c.sse" | cat -e
1$
$>./hiddenp "abc" "btarc" | cat -e
0$
$>./hiddenp | cat -e
$
$>
```

II.3 Exercice 02 - union

| | |
|---|---------------|
|  | Exercice : 02 |
| union | |
| Dossier de rendu : <i>ex02/</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.

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

Exemple :

```
$>./union zpadinton "paqefwtdjetyiytjneytjoeyjnejejj" | cat -e
zpadintoqefwjy$
$>./union ddf6vewg64f gtwthgdwthdwfteewhrtag6h4ffdhds | cat -e
df6vewg4thras$
$>./union "rien" "cette phrase ne cache rien" | cat -e
rienct phas$
$>./union | cat -e
$
$>
$>./union "rien" | cat -e
$
$>
```


II.4 Exercice 03 - ord_alphlong

| | |
|---|--|
|  | Exercice : 03 |
| | ord_alphlong |
| | Dossier de rendu : <i>ex03/</i> |
| | Fichiers à rendre : *.c, *.h |
| | Fonctions Autorisées : write, malloc, free |
| | Remarques : n/a |

Écrire un programme qui prend en paramètre une chaîne de caractères et qui affiche les mots de cette chaîne par ordre de longueur puis dans l'ordre ascii. Les majuscules et minuscules sont identiques dans l'ordre alphabétique (bien regarder les exemples). En cas d'égalité alphabétique (ex aA et Aa) les mots doivent rester dans l'ordre où ils étaient dans la chaîne d'origine (second exemple). En cas de doublons, les doublons sont conservés (dernier exemple).

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


Dans les chaînes, il n'y aura que des espaces, des tabulations et des caractères alphanumériques.

Vous n'afficherez qu'un espace entre les mots. Aucun avant le premier ni après le dernier de chaque ligne.

Exemples :

```
$>./ord_alphlong
$
$>./ord_alphlong "De son baton il frappa la pierre et l eau jaillit" | cat -e
l$
De et il la$
eau son$
baton$
frappa pierre$
jaillit$
$>./ord_alphlong "A a b B cc ca cd" | cat -e
A a b B$
ca cc cd$
$>./ord_alphlong "Pour l Imperium de l humanite" | cat -e
l l$
de$
Pour$
humanite Imperium$
$>
```

II.5 Exercice 04 - g_diam

| | |
|---|---|
|  | Exercice : 04 |
| | <code>g_diam</code> |
| | Dossier de rendu : <code>ex04/</code> |
| | Fichiers à rendre : <code>*.c</code> , <code>*.h</code> |
| | Fonctions Autorisées : <code>write</code> , <code>malloc</code> , <code>free</code> |
| | Remarques : <code>n/a</code> |


Le programme prend en paramètre une chaîne de caractères. Cette chaîne 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 représentés par des nombres et les arête par deux noeuds séparés par '-'. Par exemple, s'il existe une arête entre le noeud 2 et le noeud 3, les représentations possibles de cette arête seront soit "2-3", soit "3-2".

Le programme devra afficher la longueur du plus long chemin suivi 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.

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.6 Exercice 05 - count_island

| | |
|---|---------------|
|  | Exercice : 05 |
| | count_island |
| Dossier de rendu : <i>ex05/</i> | |
| 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 remplaçant 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.....
.....XXXXXXXX.....XXXXXXXX.....
.....XXXXXXXX.....XXX.....XXXX
.....XXXXX.....X.....XXXXXXXXXX
.....X.....
.....XXXXXXXXXX.....
.....X.....XXXXXXXXX.....
.....X.....XXXXXXXXXXXXX.....
.....X.....
XX.....XXXXX
XX.....XXXXXXXXXXXXX.....X
.....
.....X.
.....XXXX.....XX
$>
$>./count_island toto
.....00000000.....
.....00000000.....11111111.....
.....00000000.....111.....11111.....
.....000000.....2.....111111111111.....
.....2.....
.....333333333333.....2.....
.....3.....222222222.....
.....3.....222222222222.....
.....3.....
44.....5555
44.....666666666666.....5
.....
.....7.
.....88888.....77
$>
```

```
$>cat qui_est_la
.....
..X.....X.....XXXXX.....XXXXXXXXX.....XXXXXXXXXX.....
..XX.....XX.....XX.....XX.....XX.....XXXX.....XXXXXXXXXX.....
..XXXX.....XXXX.....XX.....XX.....XX.....XX.....XX.....
..XX.....XXXX.....XX.....XX.....XX.....XX.....XX.....
..XX.....X.....XX.....XX.....XXXXXXXXX.....XX.....XXXXX.....
..XX.....XX.....XXXXXXXXX.....XXXX.....XX.....XXXXX.....
..XX.....XX.....XX.....XX.....XX.....XX.....XX.....
..XX.....XX.....XX.....XX.....X.....XX.....XX.....
..XX.....XX.....XX.....XX.....X.....XXXXX.....XXXXXXXXXX.....
..XX.....XX.....XX.....XX.....XX.....XXXXXXXXXX.....XXXXX.....X.....
$>
$>./count_island qui_est_la
.....
..0.....0.....11111.....22222222.....33333333333.....4444444444.....
..00.....00.....11.....11.....22.....22.....3333.....4444444444.....
..0000.....0000.....11.....11.....22.....22.....33.....44.....
..00.....0000.....11.....11.....22.....22.....33.....44.....
..00.....0.....11.....11.....22222222.....33.....44444.....
..00.....00.....1111111111.....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.....3333333.....4444444444.....
..00.....00.....11.....11.....22.....77.....3333333333.....4444444444.....8.....
$>
```

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

II.7 Exercice 06 - Time Lord

| | |
|---|---------------|
|  | Exercice : 06 |
| Time Lord | |
| Dossier de rendu : <i>ex06/</i> | |
| Fichiers à rendre : secret | |
| Fonctions Autorisées : Tout, absolument tout ce que vous pouvez imaginer | |
| Remarques : n/a | |

Vous trouverez un exécutable nommé `time_lord` dans le répertoire `misc/` de cet examen. Quand vous exécutez 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.