



Examen

Bocal bocal@42.fr

Résumé: Ce document est votre sujet d'examen

Table des matières

| | | |
|-----------|--------------------------------------|----------|
| I | Détails administratifs | 2 |
| I.1 | Consignes générales | 2 |
| I.2 | Le Code | 3 |
| I.3 | Types d'exercices | 4 |
| II | Exercices | 5 |
| II.1 | Exercice 00 - first_word | 5 |
| II.2 | Exercice 01 - ft_range | 6 |
| II.3 | Exercice 02 - rev_wstr | 7 |
| II.4 | Exercice 03 - sort_int_tab | 8 |
| II.5 | Exercice 04 - str_maxlenoc | 9 |
| II.6 | Exercice 05 - infin_mult | 10 |
| II.7 | Exercice 06 - md5 | 11 |
| II.8 | Exercice 07 - death_race | 12 |
| II.9 | Exercice 08 - half_life_3 | 13 |

Chapitre I

Détails administratifs

I.1 Consignes générales

- Aucune forme de communication n'est permise.
- Ceci est un examen, il est interdit de discuter, d'écouter de la musique, de faire du bruit, ou de façon plus générale de produire toute 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 répertoire home contient deux dossiers : "rendu" et "sujet".
- Le répertoire "sujet" contient le sujet de l'examen. Vous avez dû le trouver, puisque vous êtes en train de lire 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. Le dépôt cessera d'accepter les pushes à l'heure précise de fin de l'examen, n'attendez donc pas le dernier moment pour pusher.
- Vous ne pouvez exécuter les programmes que vous avez compilés vous-même que dans votre dossier "rendu" et ses sous-dossiers. Cela est interdit (et d'ailleurs impossible) ailleurs.
- Chaque exercice doit être réalisé dans le répertoire 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. Le fichier auteur n'est PAS rétrovalidable.

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, et de la correction. Vous devez donc respecter les noms, les chemins, les fichiers et les répertoires...
- Les exercices stipuleront toujours les fichiers ramassés :
 - Lorsqu'un exercice demande des fichiers particuliers, ils seront nommés explicitement. Par exemple `"fichier1.c fichier1.h"`.
 - Sinon, quand les noms / le nombre de fichiers sont laissés à votre discrétion, l'exercice stipulera quelque chose de la forme `"*.c *.h"`.
 - Lorsqu'un Makefile est requis, cela sera toujours explicitement précisé.
- En cas de problème technique, de question sur le sujet, ou tout autre souci, vous devez vous lever en silence et attendre qu'un surveillant vienne à vous. Interdiction absolue de parler à vos voisins ou d'appeler oralement le surveillant.
- Tout matériel non explicitement autorisé est implicitement interdit.
- La correction ne s'arrête pas forcément au premier exercice faux. Voir la section **Types d'exercices**.
- Toute sortie de la salle est définitive.
- Un surveillant peut vous expulser de la salle sans préavis s'il le juge nécessaire.
- 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).

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. L'utilisation d'une fonction qui n'est pas autorisée est assimilée à de la triche, et sera sanctionnée par un `-42`, sans appel.
- Toute fonction non autorisée explicitement est implicitement interdite.

I.3 Types d'exercices


Il y a plusieurs types d'exercices possibles, et ils ne sont pas tous corrigés de la même façon. Voici des explications :

- **Exercice obligatoire** - Un exercice de ce type arrête immédiatement la correction s'il n'est pas réussi. Comprendre par là que vous devez absolument le réaliser si vous voulez des points pour les exercices d'après.
- **Exercice rétrovalidable** - Si vous ne rendez rien pour cet exercice, la correction ne s'arrête **PAS**, et vous pourrez obtenir les points de cet exercice quand même si vous réussissez un exercice non-bonus plus loin dans l'examen. Cependant, si vous rendez **quoi que ce soit**, et que vous échouez à l'exercice, la correction s'arrête immédiatement. Vous devez donc décider entre tenter l'exercice et risquer de perdre les points de ceux d'après, ou ne pas le tenter, et faire directement un exercice plus difficile.
- **Exercice bonus** - Un exercice de ce type n'arrête jamais la correction s'il est raté. Il ne permet pas, par contre, d'obtenir les points pour les exercices d'avant.

Chapitre II

Exercices

II.1 Exercice 00 - first_word

| | |
|---|--|
|  | Exercice : 00 |
| | first_word |
| | Dossier de rendu : <i>ex00/</i> |
| | Fichiers à rendre : first_word.c |
| | Fonctions Autorisées : write |
| | Remarques : Exercice rétrovalidable |

Écrire un programme qui prend une chaîne de caractères en paramètre, et qui affiche le premier mot de cette chaîne, suivi d'un '**\n**'.


On appelle "mot" une portion de chaîne de caractères délimitée soit par des espaces et/ou des tabulations, soit par le début / fin de la chaîne.

Si le nombre de paramètres transmis est différent de 1, ou s'il n'y a aucun mot à afficher, le programme affiche '**\n**'.

Exemple :

```
$> ./first_word "FOR PONY" | cat -e
FOR$
$> ./first_word "this      ...      is sparta, then again, maybe      not" | cat -e
this$
$> ./first_word "  " | cat -e
$
$> ./first_word "a" "b" | cat -e
$
$> ./first_word "  lorem,ipsum  " | cat -e
lorem,ipsum$
$>
```

II.2 Exercice 01 - ft_range

| | |
|---|--|
|  | Exercice : 01 |
| | ft_range |
| | Dossier de rendu : <i>ex01/</i> |
| | Fichiers à rendre : ft_range.c |
| | Fonctions Autorisées : malloc |
| | Remarques : Exercice rétrovalidable |

Écrire la fonction suivante :


```
int *ft_range(int start, int end);
```

Cette fonction doit allouer avec `malloc()` un tableau d'ints, le remplir avec les valeurs (consécutives) démarrant à **start** et finissant à **end** (**start** et **end** inclus!), et renvoyer un pointeur vers la première valeur du tableau.

Exemple :

- Avec (1, 3) vous devrez renvoyer un tableau contenant 1, 2 et 3.
- Avec (-1, 2) vous devrez renvoyer un tableau contenant -1, 0, 1 et 2.
- Avec (0, 0) vous devrez renvoyer un tableau contenant 0.
- Avec (0, -3) vous devrez renvoyer un tableau contenant 0, -1, -2 et -3.

II.3 Exercice 02 - rev_wstr

| | |
|---|---------------|
|  | Exercice : 02 |
| rev_wstr | |
| Dossier de rendu : <i>ex02/</i> | |
| Fichiers à rendre : rev_wstr.c | |
| Fonctions Autorisées : write, malloc, free | |
| Remarques : Exercice rétrovalidable | |

Écrire un programme qui prend en paramètre une chaîne de caractères, et qui affiche cette chaîne en inversant ses mots.

On appelle "mot" une portion de chaîne de caractères délimitée soit par des espaces et/ou des tabulations, soit par le début / fin de la chaîne.


Si le nombre de paramètres est différent de 1, le programme devra afficher '`\n`'.

Dans les paramètres qui seront testés, il n'y aura pas d'espaces "en trop" (comprendre par là qu'il n'y aura pas d'espaces au début ou à la fin de la chaîne, et que les mots seront toujours séparés par exactement un espace).

Exemple :

```
$> ./rev_wstr "le temps du mepris precede celui de l'indifference" | cat -e
l'indifference de celui precede mepris du temps le$
$> ./rev_wstr "abcdefghijklm"
abcdefghijklm
$> ./rev_wstr "il contempla le mont" | cat -e
mont le contempla il$
$> ./rev_wstr | cat -e
$
$>
```


II.4 Exercice 03 - sort_int_tab

| | |
|---|---|
|  | Exercice : 03 |
| | sort_int_tab |
| | Dossier de rendu : <i>ex03/</i> |
| | Fichiers à rendre : sort_int_tab.c |
| | Fonctions Autorisées : |
| | Remarques : Exercice rétrovalidable |

Écrire la fonction suivante :


```
void sort_int_tab(int *tab, unsigned int size);
```

Cette fonction doit trier (en place!) le tableau d'ints `tab`, qui contient exactement `size` entrées, dans l'ordre croissant.

Les doublons doivent être préservés.

Les entrées seront toujours cohérentes.

II.5 Exercice 04 - str_maxlenoc

| | |
|---|---|
|  | Exercice : 04 |
| | str_maxlenoc |
| | Dossier de rendu : <i>ex04/</i> |
| | Fichiers à rendre : str_maxlenoc.c |
| | Fonctions Autorisées : write, malloc, free |
| | Remarques : Exercice rétrovalidable |

Écrire un programme qui prend en paramètres n chaînes de caractères et qui affiche, suivie d'un retour à la ligne, la plus grande chaîne de caractères incluse dans toutes les chaînes passées en paramètres. Si plusieurs chaînes correspondent, on affichera celle qui apparaît en premier dans le premier paramètre. A noter que "" est forcément dans toutes les chaînes.


Si aucun paramètre n'est transmis, le programme doit afficher '\n'.

Soit A et B des chaînes de caractères. On dit que A est inclus dans B si A est une sous-chaîne 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 afkuonjosurafig headfgllof fghellosag afdfbosnjourafg
os
$>./str_maxlenoc | cat -e
$
```

II.6 Exercice 05 - infin_mult

| | |
|---|---------------|
|  | Exercice : 05 |
| infin_mult | |
| Dossier de rendu : <i>ex05/</i> | |
| Fichiers à rendre : *.c, *.h | |
| Fonctions Autorisées : write, malloc, free | |
| Remarques : Exercice rétrovalidable | |

Écrire un programme qui prend en paramètres deux chaînes de caractères représentant des nombres de longueur potentiellement infinie, et affiche sur la sortie standard le résultat de la multiplication de ces deux nombres, suivi d'un '\n'.


Un nombre négatif sera précédé d'un et un seul signe -. Les seuls caractères qui feront potentiellement partie de ces chaînes sont les chiffres et le signe -.

Tous les paramètres seront bien formatés, et il y a toujours exactement deux paramètres, pas de pièges.

Exemple :

```
$> ./infin_mult "879879087" "67548976597" | cat -e
59434931855952726939$
$> ./infin_mult "-876435" "987143265" | cat -e
-865166907460275$
$> ./infin_mult "-807965" "-34532"
27900647380
$>
```

II.7 Exercice 06 - md5

| | |
|---|---------------|
|  | Exercice : 06 |
| md5 | |
| Dossier de rendu : <i>ex06/</i> | |
| Fichiers à rendre : *.c, *.h | |
| Fonctions Autorisées : write, malloc, free | |
| Remarques : Exercice rétrovalidable | |

Écrire un programme qui prend une chaîne de caractères en paramètre, et qui affiche le digest md5 de cette chaîne, suivi d'un '\n'.


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

Nous vous fournissons en annexe la RFC md5 que vous devez utiliser.

Exemple :

```
$> ./md5 "" | cat -e
d41d8cd98f00b204e9800998ecf8427e$
$> ./md5 "ceci n'est pas une pipe" | cat -e
abec1071c807ebfec3045ce68ce0df47$
$> ./md5 "jamais de la vie je ne ferai ca dans mon sandwich ..." | cat -e
94f1547ed3f035b2baf450bc8e6516bb$
$> ./md5 "le bocal est beau" | cat -e
69b5af91d872e9b03cd76eb6d6a5bef6$
$> ./md5 "le bocal est" "fort" | cat -e
$
$>
```


II.8 Exercice 07 - death_race

| | |
|---|--|
|  | Exercice : 07 |
| | death_race |
| | Dossier de rendu : <i>ex07/</i> |
| | Fichiers à rendre : secret |
| | Fonctions Autorisées : Tout ce que vous voulez |
| | Remarques : Exercice bonus |

Vous trouverez dans l'annexe du sujet un exécutable **death_race** ainsi que sa source (censurée), **race.c**.

Vous devez rendre un fichier **secret** contenant la phrase secrète qui vous est donnée par l'exécutable **death_race** fourni, sans aucun caractère ou saut de ligne supplémentaire.

II.9 Exercice 08 - half_life_3

| | |
|---|--|
|  | Exercice : 08 |
| | half_life_3 |
| | Dossier de rendu : <i>ex08/</i> |
| | Fichiers à rendre : secret |
| | Fonctions Autorisées : Tout ce que vous voulez, y compris les prieres vaudou |
| | Remarques : Exercice bonus |

Vous trouverez en annexe de ce sujet un binaire `half_life_3`. Vous devez l'activer, et pour ça, il vous faut une clé.

Vous devez rendre un fichier **secret** contenant une clé acceptée par ce binaire, sans saut de ligne ni caractère supplémentaire.