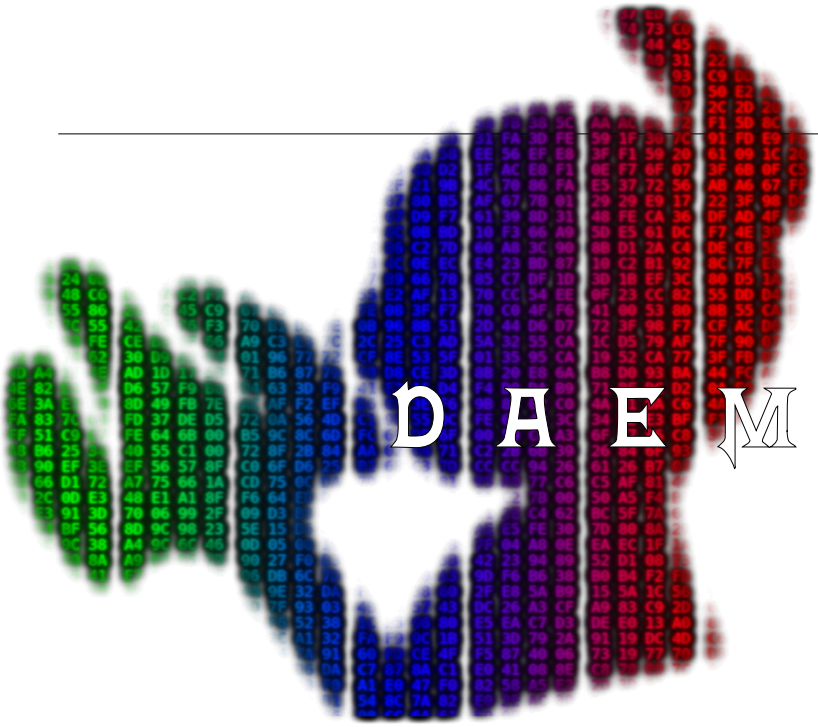


DAEMONLAB

EXAMEN

theme HOSPITAL™

Analyse grammaticale

- DaemonLab -

Vous trouverez la bande son du jeu en ressources de l'examen.
Écoutez la en faisant l'examen, c'est obligatoooaAAAreee…
Enfin, c'est comme vous voulez quoi…

Ce document est strictement personnel et ne doit en aucun cas être diffusé.

INDEX

01 – Avant-propos
02 – Fonctions autorisées

03 – Nettoyage
04 – Unique
05 – Extrait
06 – Trie

01 – Avant-propos

Votre travail doit être rendu via le dossier ~/exam/hospital/ dans votre espace personnel.

Si vous faites erreur et que le dossier que vous utilisez pour votre rendu est différent, vous ne serez pas évalue faute d'avoir pu trouver votre travail.

Ce travail est a effectuer seul. De plus il s'agit d'un examen. Vous n'avez donc pas le droit de communiquer avec vos camarades.

Votre rendu doit respecter strictement l'ensemble des règles suivantes :

- Il ne doit contenir aucun fichier objet. (*.o)
- Il ne doit contenir aucun fichier tampon. (*~, #*#)
- Il ne doit pas contenir votre production finale (programme ou bibliothèque)

La présence d'un fichier interdit mettra immédiatement fin a votre évaluation !

Votre programme doit respecter les Tables de la Norme dans leur intégralité. Vous êtes invité à les observer depuis l'Infosphère. Elles sont disponibles comme ressource de cette activité.

02 – Fonctions autorisées

La bibliothèque logicielle venant avec le C est vaste et disponible. La LibLapin, que vous utilisez dans vos projets multimédia, est également vaste… Cependant nous avons fait le choix de vous interdire leurs utilisation intégrales, afin de vous amener progressivement à reprogrammer vous même ses fonctionnalités les plus utiles.

L'utilisation d'une fonction interdite est assimilée a de la triche. La triche provoque l'arrêt de l'évaluation et la perte des médailles.

Vous n'avez le droit d'utiliser aucune fonction issue de la LibC ou de la LibLapin à l'exception de celles que nous vous autoriserons explicitement.

Pouvoir utiliser une fonction ne signifie pas nécessairement que celle-ci soit utile a votre cas.

Pour cette activité, issu de la LibC, vous n'avez le droit qu'à la liste suivante :

- open
- close
- read
- srand
- cos
- atan2
- time
- write
- alloca
- atexit
- rand
- sin
- sqrt
- malloc, free

03 – Nettoyage

Fichier clean_string.c

Garder l'hopital propre est la premiere des missions de l'hopital, tout comme nettoyer vos instruments est la premiere des obligations en medecine ! Je vous l'ai deja dit des centaines de fois... Vous le voulez pas encore tuer votre patient, non ? Vous avez de la chance qu'on ait interdit aux patients de mourir dans les couloirs...

Implémentez la fonction suivante :

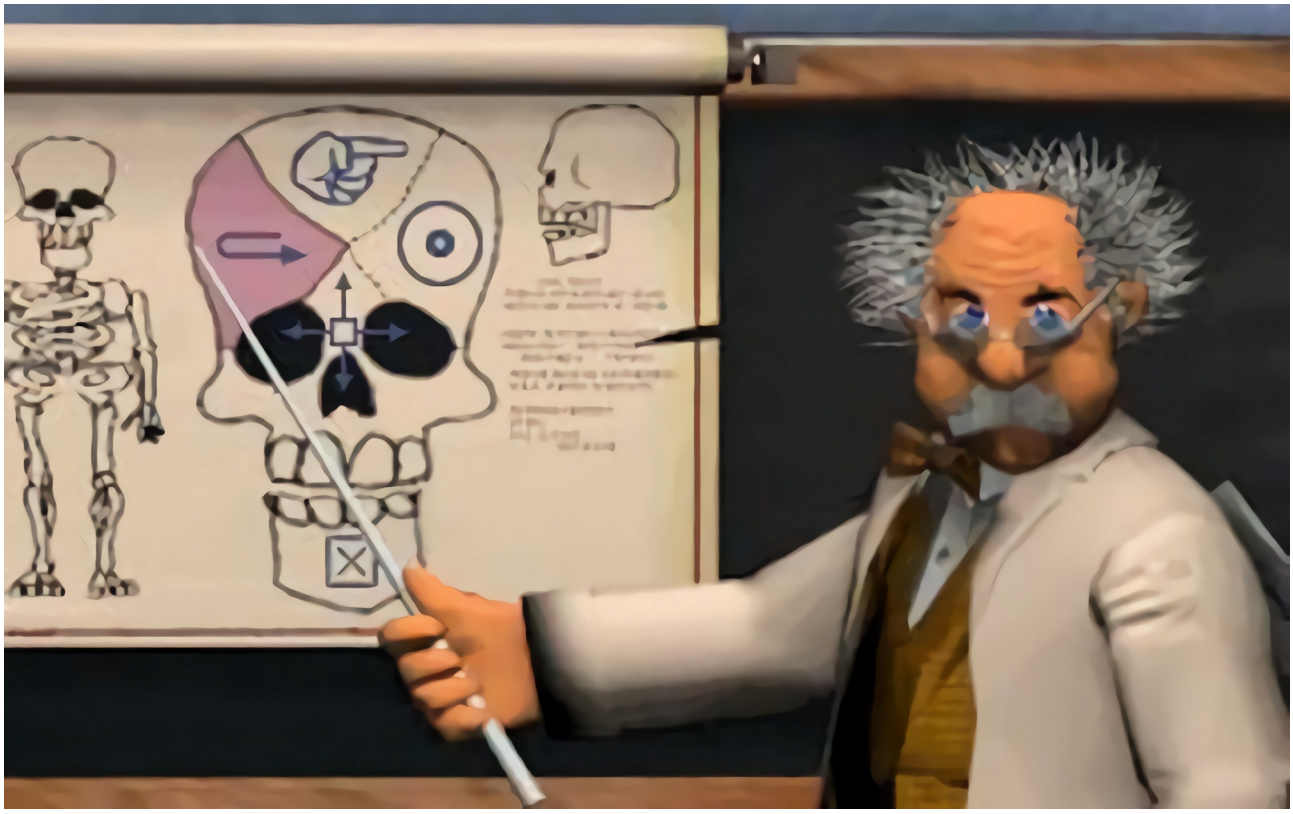
```
char        *std_clean_string(const char        *str)
```

Cette fonction génère une nouvelle chaîne de caractères où tous les espaces, tabulations, saut de lignes et tabulations verticales en début et en fin de chaîne de caractère ont été retirés. Elle réduit également tout présence de ces caractères entre les symboles n'étant pas ces caractères à un seul et unique espace.

Par exemple, " a b " deviendra "a b".

04 – Unique

Fichier unique.c, unique.h



Un seul exemplaire de chaque ouvrage est suffisant pour apprendre. Un seul dossier par patient suffira donc, quelque soit son historique. Pourquoi garder une trace du nombre de fois ou ils reviennent, de leurs problemes du passe ? Qu'est ce que cela nous apprend ? Rien !

Implémentez la fonction **std_unique** :

```
bool            std_unique(t_unique      *uniq,
                           const char    *str);
```

La fonction **std_unique** exploite la structure t_unique que vous devrez définir dans unique.h et que vous êtes libre de former comme vous le souhaitez, ainsi qu'une chaîne de caractère.

L'objectif de **std_unique** est de renvoyer **vrai** ou **faux** si le dernier appel à **std_unique** à été réalisé avec une chaîne contenant les mêmes caractères que **str**.

05 – Extrait

Fichier extract.c

Il arrive parfois qu'un client... pardon qu'un patient doive subir une ablation. L'ablation est une operation comme une autre : l'acte de guerir passe parfois par le retrait d'une jambe ou deux... L'important n'est il pas de preserver la vie ?

Implémentez la fonction suivante :

```
int                 std_extract(char        *str,
                                size_t      index,
                                ssize_t     size)
```

Cette fonction modifie **str** en retirant **size** caractères à partir de **index**. Le terminateur n'est **jamais** retiré. Le nombre de caractères à retirer **size** peut-être négatif. Dans ce cas, au lieu d'aller vers la fin de la chaîne de caractère, la suppression s'opère vers le début. On ne peut pas retirer de caractères avant le premier.

La fonction renvoi le nombre de caractères retirés.

06 – Trie

Fichier sort.c

Le triage est un moment extremement important lors de l'arrive de clients aux urgences. Certains arriveront avec des blessures graves, d'autres parfois plus impressionnantes mais moins lethale. En tant que responsable de la communication de l'hopital, mon role est d'amener l'equipe soignante a privilegier les soins des blessures impressionnantes, et celui du superviseur de favoriser les clients dotes d'une assurance maladie.

Implémentez la fonction suivante :

```
char            **std_split_sort(const char   *s,
                                 char         separator)
```

Cette fonction alloue un tableau de chaîne de caractères contenant les caractères situés entre les instances du séparateur passé en paramètre. Par exemple, "abc,def,ghi" avec virgule comme séparateur va renvoyer un tableau de 4 cases contenant les trois chaînes suivi d'un pointeur NULL.

Les éléments du tableau renvoyé seront triés par ordre ASCII.