D A E M O N L A B

Journée 5

Analyse grammaticale, arithmétique des pointeurs

- DaemonLab -

Cette journée est la dernière avant la première évaluation de C. Vous n'y découvrirez rien de neuf mais travaillerez des fonctions plus complexes que précédemment.

INDEX

01 – Avant-propos

02 – Fonctions autorisées

03 – Exercices principaux

04 – Aller plus loin

01 – Avant-propos

Votre travail doit être rendu via le dossier **~/hyperespace/j5/** dans votre espace personnel.

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

Ce travail est a effectuer seul. Vous pouvez bien sur échanger avec vos camarades, néanmoins vous devez être l'auteur de votre travail. Utiliser le code d'un autre, c'est **tricher**. Et tricher annule **toutes** les médailles que vous avez reçu sur l'activité. La vérification de la triche est réalisée de la même manière que la correction : de manière **automatique**. Prenez garde si vous pensez pouvoir passer au travers.

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 à 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. Nous avons cependant fait le choix de vous interdire son utilisation, afin de vous amener progressivement à reprogrammer vous même ses fonctionnalités les plus utiles.

L'utilisation d'une fonction interdite est assimilée à 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 à l'exception de celles que nous vous autoriserons explicitement.

Pour cette activité, vous avez le droit à l'appel système **write** ainsi qu'aux fonctions **malloc** et **free**.

03 – Exercices principaux

Programmez les fonctions **std_fonction** et **test_fonction**, situé dans les fichiers **fonction.c** pour chacune des fonctions situés ci-dessous. « **fonction** » est à remplacer par le nom de la fonction que vous recodez.

```
void swap(int *a, int *b);
char *trim(const char *str);
size_t strspn(const char *str, const char *accept);
int strtol(const char *str, const char **e, int base);
int print_base(int n, const char *base);
int sqrt(int nbr);
double powf(double x, double y);
```

La fonction **swap** intervertit les entiers contenu par ***a** et ***b**.

La fonction **trim** alloue la place pour une nouvelle chaîne de caractère qui sera remplie en tirant des éléments de **str**. Seule différence : **trim** retirera tous les caractères d'espacement (**isspace**) qui sont situés au début et à la fin de la chaîne de caractère.
Exemple : " abc " générera "abc".

Vous pouvez trouver les fonctions **strspn, strtol** dans leurs manuels respectifs.

La fonction **print_base** affiche **n** à l'aide de la base passée en paramètre. Par exemple, si base vaut "0123456789", l'entier sera affiché en base 10, et exploitera les chiffres passés en paramètre. Si la base vaut "ABC", l'entier sera affiché en base 3, et utiliser A pour 0, B pour 1 et C pour 2.

Les deux dernières fonctions sont également standard et disposent d'un manuel. La fonction **powf** est une version de **pow** gérant les nombres à virgule.

04 – Aller plus loin

Programmez les fonctions std_fonction et test_fonction, situé dans les fichiers fonction.c pour chacune des fonctions situés ci-dessous :

```
float strtod(const char *tok, const char **e);
char *strlcat(char *target, const char *src, size_t l);
size_t strcspn(const char *str, const char *reject);
char *strpbrk(const char *str, const char *accept);
char *strncat(char *t, const char *s, size_t len);
```