D A E M O N L A B

EXAM A/A

Examen de difficulté A

- DaemonLab -

INDEX

01 – Avant-propos
02 – Fonctions autorisées

03 – Saut de ligne
04 – Distance
05 – 10 fois
06 – Lettre
07 – Triangle
08 – PunchCard

01 – Avant-propos

Votre travail doit être rendu via le dossier **~/hyperespace/exam0/** 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. 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 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. Dans le cadre de cette activité, la seule fonction à laquelle vous aurez le droit est l'appel système write.

Vous pouvez observer le fonctionnement de write en tapant « man 2 write » dans votre terminal, néanmoins, afin de vous faciliter son utilisation au départ, nous vous avons préparé une fonction que vous êtes libre d'intégrer à votre code.

Cette fonction est tc_putchar, elle tente d'écrire le caractère passé en paramètre sur la sortie standard (le terminal, par défaut). Elle renvoi 1 si elle a réussie à écrire le caractère, sinon elle renvoi 0. Ci-dessous, le code de tc_putchar suivi d'un programme de test.

```
#include <unistd.h>

int tc_putchar(char c)
{
     return (write(1, &c, 1) > 0);
}
```

```
int main(void)
{
     tc_putchar('A');
     tc_putchar('\n');
     return (0);
}
```

03 – Saut de ligne

newline.c

Écrire un programme qui effectue un saut de ligne.
Votre programme devra renvoyer 0.

04 – Distance

distance.c

Implémentez la fonction suivante :

```
int                 std_distance(int        a,
                                 int        b)
```

Cette fonction renvoi la distance entre **a** et **b**. Cette distance est toujours positive.

05 – Dix fois

ten.c

Écrivez un programme qui affiche 10 fois le caractère 'z'.
Votre fonction **main** ne devra pas dépasser 9 lignes.
Votre programme devra retourner 50.

06 – Letter

letter.c

Implémentez la fonction suivante :

```
int             std_display_letter(int      nbr)
```

Cette fonction affiche une lettre. Cette lettre sera en minuscule si **nbr** est compris entre 0 et 25, en partant de 'a'. Si **nbr** est compris entre 26 et 51, il affichera la lettre en majuscule.
Si la fonction a affiché un caractère, elle renverra 1, sinon elle renverra 0.

07 – Triangle

triangle.c

Implémentez la fonction suivante :

```
void            std_write_triangle(int        nbr)
```

Cette fonction affiche un triangle composé du caractère '-' dans le terminal. Ci-dessous, un exemple de triangle de taille 5.

```
$> ./a.out
-----
----
---
--
-
$>
```

08 – Punch Card

punchcard.c

Implémentez la fonction suivante :

```
void            std_punch_card(char            *nbr,
                               int             line,
                               int             column)
```

Cette fonction prend en paramètre une « carte perforée », composée exclusivement de '-' et saut de ligne. La fonction std_punch_carc place un espace ' ' à la ligne line et à la colonne column. La taille de la carte ne vous est pas transmise, donc vous devez la déduire. Vous avez la garantie qu'elle est rectangulaire.

```
$> ./a.out
------
- ----
------
------
---- -
------
- ----
--- --
------
$>
```