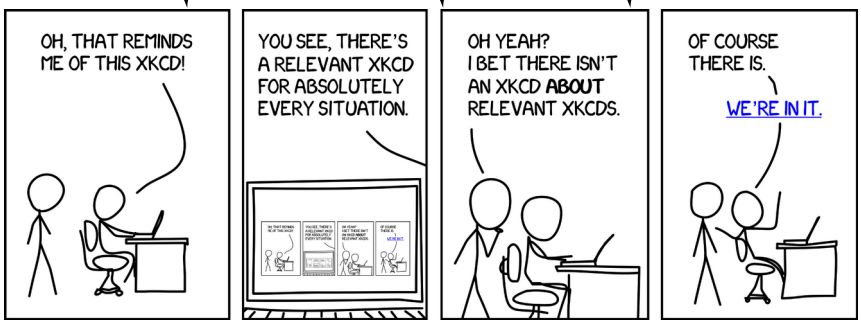
DAEMONLAB

RECURSOR I



- DaemonLab -
daemonlab@ecole-89.com

Ce TP aborde la sujet des fonctions qui s'appellent elle-même.
Concernant la bande dessinée, xkcd, c'est bon, mangez en.

Nom de code : !recursor1
Clôture du ramassage : 21/01/2020 23:59

Médailles accessibles :

Définition des médailles à venir

INDEX

Avant-propos :

01 – Détails administratifs

Votre travail doit être envoyé via l'interface de ramassage du **TechnoCentre** :

http://technocentre.ecole89.com/ramassage

Le numéro de code présent sur ce sujet vous est propre : vous devrez le renseigner en rendant votre travail. En cas d'erreur, votre travail ne sera pas associée à l'activité et votre travail ne sera pas ramassé.

Pour cette activité, vous rendrez votre travail sous la forme d'une archive au format tar.gz. Cette archive devra contenir l'ensemble de votre travail tel que demandé dans la section 4.

Pour créer cette archive .tar.gz, il vous suffit d'utilliser la commande suivante :

```
$> tar cvfz mon_fichier.tar.gz fichier1 fichier2 fichier3
```

Le nom « mon_fichier.tar.gz » étant à remplacé par le nom que vous souhaitez donner votre fichier archive, et « fichier1 », « fichier2 », « fichier3 » par les fichiers ou dossiers que vous souhaitez mettre dans cette archive. Vous pouvez vérifier le contenu de votre archive à l'aide de la commande « **`tar -t mon_archive.tar.gz`** ».

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.

Médailles accessibles :

Réussir à rendre :
Vous avez réussi à envoyer votre travail au système de correction.

02 – Propreté de votre rendu

Votre rendu, c'est à dire le contenu de l'archive ou du dépôt que vous entrez sur l'interface du TechnoCentre, doit respecter **strictement** l'ensemble des règles suivantes :

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

La présence d'un fichier interdit mettra immédiatement fin à votre évaluation.

Médailles accessibles :

Rendu propre
Votre rendu respecte les règles de propretés imposées.

03 – Travail

Écrivez la fonction suivante :

```
int             e89_recpow(double          x,
                           int             y)
```

Cette fonction fonctionne exactement comme la fonction **pow** de la bibliothèque standard ou comment votre **e89_pow**, réalisé lors de votre période d'apprentissage. Il y a néanmoins une différence d'implémentation de taille.

Implémentez **e89_recpow** sans utiliser de boucle. Les mot-clefs **white** et **do** sont interdits, en plus de l'habituelle interdiction de **for**.

Comment faire ? A quoi sert votre boucle ? A faire progresser un entier depuis 0 jusqu'à y. Pourriez vous à la place jouer avec un appel à la fonction **e89_recpow** situé dans cette même fonction et avec ses paramètres ?

Attention spoiler : la réponse est oui. Trouvez comment.

Écrivez la fonction suivante :

```
int             e89_recfactoriel(int       value);
```

De la même manière qu'avec **e89_pow** et **e89_recpow**, la fonction **e89_recfactoriel** ne doit pas utiliser de boucle. Elle calcule la factorielle de **value**.

De la même manière qu'avec **e89_factoriel**, si le nombre à calculer est trop grand, la fonction renverra -1 et mettra dans **errno** une valeur appropriée.

Écrivez la fonction suivante :

```
int             e89_recfibonacci(int      value);
```

La fonction **e89_recfibonacci** calcule la valeur numéro **value** dans la suite de Fibonacci caractérisée de la façon suivante :

f(x) = f(x – 1) + f(x – 2)
f(0) = 0
f(1) = 1

Cette fonction calculera la valeur en utilisation la récusion, c'est à dire en s'appelant elle-même. Cette façon de faire est dite naïve car elle comporte certains défauts. Lesquels ? Tentez de faire appel à un très grand nombre.

Écrivez la fonction suivante :

```
int             e89_ackermann(int         m,
                              int         n);
```

Correspondant aux formules suivantes :

ackermann(m, n) =
- **Si m = 0**, n + 1
- **Si m > 0 et n = 0**, ackerman(m – 1, 1)
- **Si m > 0 et n > 0**, ackermann(m – 1, ackermann(m, n - 1))