TD 18 : Figures alphanumériques

Exercice 1 (Représentation des entiers relatifs en complément à deux)

1. Comment représenter -538 sur 16 bits ?
2. Ecrire une fonction `complement_a_deux(n,N)` qui prend en argument un entier relatif `n` et un entier naturel `N` et qui retourne la représentation de `n` en complément à deux sur `N` bits, sous la forme d'une chaîne de caractère.
3. Ecrire une fonction `c2N_to_decimal(ch,N)` qui prend en argument une chaîne de caractère `ch` et un entier naturel `N` et qui retourne l'entier relatif `n` dont `ch` est la représentation en complément à deux sur `N` bits.

Rappel : 'ab'*3 renvoie 'ababab' (concaténation multiple).
La suite de ce TD est à la fois simple mais très importante à comprendre pour ce qui arrive après (notamment le parcours de graphe). Vous allez étudier des appels récursifs et il vous faut être capable de déterminer la figure alphanumérique obtenue en fonction de la **position** de l'appel récursif.

```python
def triangle1(n) :
    print('* '*n)
    if n > 0 :
        triangle1(n-1)
```

```python
def triangle2(n) :
    if n>0 :
        triangle2(n-1)
    print('* '*n)
```

Exercice 2 (Questions de triangle)

1. Comparer les deux fonctions `triangle1` et `triangle2`. Contiennent-elles les mêmes éléments ? Dans le même ordre ?
2. Prédire la figure obtenue pour chaque fonction.
3. Tester vos prédictions !
4. Rédiger une conclusion.

```
* * *
* *
*
```

Figure 1

```
*
* *
*
* * *
*
* *
*
```

Figure 2

```
=====     ====================
====     ===================
===     ==================
==     =================
=     ================

=     ================
==     =================
===     ==================
====     ===================
=====     ====================
```

Figure 3

```
mmmmmmmmmmmmmmmmmmmmm
m_________________m
m_________________m
m_________________m
m_________________m
m_________________m
m_________________m
m_________________m
m_________________m
m_________________m
mmmmmmmmmmmmmmmmmmmmm
```

Figure 4

Exercice 3 (Figures)

Reproduire les figures ci-dessus. La fonction `figure4` doit permettre de choisir les 2 caractères utilisées pour la figure ainsi que la taille `n` de la figure.
Si vous avez le temps et de l'imagination, proposez votre figure !
Lors du dépôt du programme sur `infogl.free.fr`, il faut que toutes vos figures soient visibles lors de l'exécution de votre programme.

Exercice 4 (Représentation d'un nombre réel en informatique)

Un nombre réel

$$\pm \texttt{1.xxxxx} \times 2^n \tag{1}$$

est représenté en mémoire (selon la norme IEEE754) sur 64 bits avec dans l'ordre 1 bit de signe (`0` $\rightarrow$ positif), 11 bits pour l'exposant ($n + 1023$) et 52 bits pour la partie décimale `xxxxx` de la mantisse (ne pas oublier le 1 devant !).
Vous avez appris à convertir un nombre entier en base 2, on continue de faire de même : on développe la représentation binaire sur les puissances positives de 2, puis sur les puissances négatives de 2 pour la partie décimale à représenter en base 2. Une manière de faire est la suivante :

Convertir un réel $x < 1$ en base b
- On multiplie x par b
- On lit le chiffre des unités de x (c'est sa partie entière)
- On retire à x sa partie entière et on recommence

1. Montrer que 19.2 se convertit en 10011.00110011... Pour trouver sa représentation en mémoire, il faudrait encore l'écrire sous la forme (1). Que remarquez-vous sur la partie venant après la virgule ?
2. Appliquer la méthode à 0.1, puis 0.3. Ces valeurs sont-elles représentées de manière exacte en base 2 ?
3. Dans l'interprêteur python, tester `1+1+1 == 3`, puis `0.1+0.1+0.1 == 0.3`. Expliquer.
4. Est-il judicieux de faire des tests sur des flottants ? Par quoi est-il préférable de remplacer ces tests ?
5. L'associativité des réels est-elle préservée en informatique ?