Devoir maison n°1

Exercice 1. Informatique I (CCP 2013 modifié)

On munit le plan d'un repère orthonormé. On considère la conique $\mathcal{H}$ d'équation cartésienne :

$$x^2 - 13y^2 = 1$$

1. Tracer l'allure de l'hyperbole $\mathcal{H}$. On précisera sur le dessin les tangentes aux points d'ordonnée nulle ainsi que les branches infinies.
2. Écrire un algorithme en Python qui renvoie sous la forme d'une liste les éventuels couples d'entiers **naturels** (x, y) vérifiant :

$$\begin{cases} x^2 - 13y^2 = 1 \\ y \leq 200. \end{cases}$$

On rappelle qu'en Python la fonction `sqrt` *peut être obtenu grâce au module* `math` *et que la fonction partie entière est appelée* `int`*.*

Exercice 2. Informatique II (CCP 2015)

Les algorithmes demandés doivent être écrits en Python. On sera très attentif à la rédaction et notamment à l'indentation du code.

Voici, par exemple, un code Python attendu si l'on demande d'écrire une fonction nommée `maxi` qui calcule le plus grand élément d'un tableau d'entiers :

```
def maxi(t):
    """Données: t un tableau d'entiers non vide
       Résultat: le maximum des éléments de t"""
    n =len(t) # la longueur du tableau t
    maximum = t[0]
    for k in range(1,n):
        if t[k] >  maximum:
            maximum = t[k]
    return maximum
```

L'instruction `maxi([4,5,6,2])` renverra alors 6.

I.1. Donner la décomposition binaire (en base 2) de l'entier 21.

On considère la fonction `mystere` suivante :

```
def mystere(n, b):
    """Données: n > 0 un entier et b > 0 un entier
       Résultat: ......."""
    t = [] # tableau vide
    while n > 0:
        c = n % b
        t.append(c)
        n = n // b
    return t
```

On rappelle que la méthode `append` rajoute un élément en fin de liste. Si l'on choisit par exemple `t = [4,5,6]`, alors, après avoir exécuté `t.append(12)`, la liste `t` a pour valeur `[4,5,6,12]`.

Pour $k \in \mathbb{N}^*$, on note c_k, t_k et n_k les valeurs prises par les variables `c`, `t` et `n` à la sortie de la k-ème itération de la boucle "while".

I.2. Quelle valeur est renvoyée lorsque l'on exécute `mystere(256,10)` ?

On recopiera et complétera le tableau suivant, en ajoutant les éventuelles colonnes nécessaires pour tracer entièrement l'exécution.

k	1	2	$\cdots$
c_k			$\cdots$
t_k			$\cdots$
n_k			$\cdots$

I.3. Soit $n>0$ un entier. On exécute `mystere(n,10)`. On pose $n_0=n$.

I.3.a. Justifier la terminaison de la boucle `while`.

I.3.b. On note p le nombre d'itérations lors de l'exécution de `mystere(n,10)`. Justifier que pour tout $k\in[\![0,p]\!]$, on a $n_k\leq\frac{n}{10^k}$. En déduire, une majoration de p en fonction de n.

I.4. En s'aidant du script de la fonction `mystere`, écrire une fonction `somme_chiffres` qui prend en argument un entier naturel et renvoie la somme de ses chiffres. Par exemple, `somme_chiffres(256)` devra renvoyer 13.

I.5. Ecrire une version récursive de la fonction `somme_chiffres`, on la nommera `somme_rec`.

Exercice 3. Informatique III

Dans cet exercice, les algorithmes demandés doivent être écrits en Python. On sera très attentif à la rédaction et notamment à l'indentation du code.

1. On considère la fonction Python suivante qui, pour un entier n passé en argument, renvoie la somme $\sum_{i=0}^{n} i^2$:

```python
def somme_carres(n):
    """ Somme des entiers compris entre 0 et n chacun élevé au carré """
    S = 0
    for i in range(n+1):
        S = S+i**2
    return S
```

   a. Déterminer la complexité (temporelle) de cet algorithme en terme d'opérations élémentaires (additions, puissances)

   b. Écrire en Python une version récursive de l'algorithme précédent.

2. On considère la fonction Python suivante qui, pour un entier n passé en argument, renvoie le terme d'indice n d'une suite $(u_n)_{n\in\mathbb{N}}$:

```python
def suite(n):
    """ Calcule le terme d'indice n de la suite """
    if n == 0:
        return 2
    return 2*suite(n-1)-1
```

a. Donner l'expression sous forme d'une suite récurrente $\begin{cases} u_0 = ? \\ u_{n+1} = ? \end{cases}$ de la suite (u_n) représentée par l'algorithme `suite(n)` précédent.

b. Déterminer la complexité (temporelle) de cet algorithme en terme d'opérations élémentaires (additions, soustractions, multiplications).

c. Écrire une version itérative de cet algorithme.

d. On considère la fonction Python suivante :

```python
def mystere(N):
    """ Fontion mystere """
    n = 0
    while suite(n)<=N:
        n=n+1
    return n
```

Que détermine la fonction `mystere` (qui utilise la fonction `suite` définie plus haut) appliquée à un entier `N` ?