Institut Préparatoire aux Etudes d'Ingénieur de Sfax : AU 2015-2016
Durée : 1h - Classes : MP1-PC1-PT1

Devoir d'informatique
Février 2016

Les programmes doivent être écrits d'une manière parfaitement lisibles, avec une indentation convenable et en respectant les notations introduites. Ils doivent être documentés par des explications concises et précises.

Barème : 5 points par exercice.

Exercice 1 : Calcul de la racine carrée entière d'un entier positif

Pour calculer la racine carrée entière d'un entier positif nous proposons d'effectuer des soustractions successives des nombres impairs. En effet, si p est l'entier solution de la double inéquation :

$$\sum_{i=1}^{p}(2i-1) \leq n < \sum_{i=1}^{p+1}(2i-1)$$

alors

$$p \leq \sqrt{n} < p+1$$

et donc p est la racine carrée entière de n.

Exemple : Pour calculer la racine carrée entière de 43, on procède de la manière suivante : $43-1=42,\ 42-3=39,\ 39-5=34,\ 34-7=27,\ 27-9=18,\ 18-11=7$

La prochaine soustraction $7-13$ donnerait un résultat négatif. En total nous avons effectué 6 soustractions, donc la racine carrée entière de $43 = 6$.

Écrire en Python une fonction récursive renvoyant la racine carrée entière d'un entier positif n passé en paramètre.

Exercice 2

Écrire en Python une fonction itérative d'en-tête ***dict_ch(chaine)*** renvoyant le dictionnaire du nombre d'occurrences des caractères pour la chaîne passée en paramètre.

- Exemple :

  L'appel ***dict_ch('programme')*** retourne le dictionnaire suivant :

  $\{'a' : 1, 'r' : 2, 'm' : 2, 'e' : 1, 'o' : 1, 'p' : 1, 'g' : 1\}$.

Exercice 3

Le nombre de combinaisons C_n^p représente le nombre de sous-ensembles de cardinal p d'un ensemble de cardinal n. Il est défini par :

$$C_n^p = \begin{cases} 1 & si \quad p = 0 \; ou \; p = n \\ \dfrac{n * C_{n-1}^{p-1}}{p} & sinon \end{cases}.$$

Soit la fonction ***combRec(n,p)*** où n et p sont deux entiers positifs tels que $p < n$.

```
def combRec ( n , p ) :
    if( p  ==  0 or n  ==  p ) :
        return(1)
    else:
        return(n* combRec ( n-1,p-1)//p)
```

Faire le tournage à la main de la fonction ***combRec*** pour l'appel suivant : ***combRec(6,4)***.

Exercice 4

Soit la fonction Python suivante :

```
def consecR(L,c,m):
    if len(L)==0:
        return(m)
    if L[0]==0:
        return(consecR(L[1:],c+1,max(m,c)))
    else:
        return(consecR(L[1:],0,max(m,c)))
```

1. Que renverra l'appel suivant :

   ```
   print(consecR([1,0,0,2,0,0,0,0,3,0],0,0))
   ```

2. Proposer une version itérative ***consecI(L)*** de la fonction ***consecR***.