La récursivité

Mini Projet N°19

A. Soient a et n deux entiers naturels non nuls et **Puis** une fonction qui calcule **a** à la puissance **n** (**a^n**) définie de la façon suivante :

Puis(a,0) = 1
Puis(a,n) = Puis(a*a, n div 2) *Si **n** est **pair***
Puis(a,n) = a * Puis(a*a, (n-1) div 2) *Si **n** est **impair***

Que remarquez-vous de la façon dont la fonction **Puis** est définie ? Qu'appelle-t-on ce procédé ?

………

………

B. On se propose d'écrire un programme qui permet de :

- Remplir un fichier « **Nombre.txt** » par des couples de valeurs (a,n), a et n sont des entiers strictement positifs.
- Stocker dans un fichier texte nommé « **puissance.txt** » les couples de valeurs (a,n) suivies par la valeur de **a** à la puissance **n**, tel que : chaque ligne du fichier contient un couple de valeur (a,n) suivi par le symbole « = », suivi par la valeur a^n.

Travail demandé :

1) Ecrire l'algorithme du programme principal en le décomposant en modules.
2) Ecrire l'algorithme de chaque module envisagé.
3) Implémenter la solution en Python.

La récursivité est une méthode de résolution des problèmes algorithmiques qui consiste à appeler un sous-programme dans son propre corps.

Un sous-programme récursif est un module qui **s'appelle lui-même** avec d'autres paramètres jusqu'à ce qu'une condition d'arrêt soit atteinte.

Forme générale:

Entête du module avec présence obligatoire de(s) paramètre(s)
Début
« instructions »
Si *CONDITION_D'ARRET* **Alors**
« instruction du point d'arrêt »
Sinon *{cas général}*
« Appel au module récursif avec **changement obligatoire de(s) paramètre(s)**»
FinSi
Fin

Point d'arrêt

Appel récursif du module
Au moins un paramètre doit être changé à chaque appel.

Remarque

Comment concevoir un sous-programme récursif ?

Dans l'écriture des programmes récursifs on retrouve généralement les étapes suivantes :

1. *Trouver une décomposition récursive du problème*
   (a) Trouver l'élément de récursivité qui permet de définir les cas plus simples (*ex.* une valeur numérique qui décroît, une taille de données qui diminue).
   (b) Exprimer la solution dans le cas général en fonction de la solution pour le cas plus simple.
2. *Trouver la condition d'arrêt de récursivité et la solution dans ce cas*
   – Vérifier que la condition d'arrêt est atteinte après un nombre fini d'appels récursifs dans tous les cas
3. *Réunir les deux étapes précédentes dans un seul programme*

Exercice 1 :

Soit l'algorithme de la fonction **Inconnu** suivant :

```
Fonction Inconnu  (n : entier) : entier
Début
     Si (n=0) Alors
          Retourner 0
     Sinon
          Retourner  n + Inconnu (n-1)
     Fin Si
Fin
```

Questions :

1) Quel est le résultat retourné par la fonction **Inconnu** pour **n=5**.
2) Déduire le rôle de cette fonction ?

Exercice 2 :

Soit l'algorithme de la fonction **Inconnu** suivant :

```
Fonction Inconnu ( …………………………….) : ……………………………….
Début
     Si Long(ch)=0 Alors
          Retourner 0
     Sinon
        Si ch[Long(ch)-1] ∈ ["0".."9"] Alors
            d←Valeur (ch[Long(ch)-1])
            Retourner d + Inconnu (Sous-chaine (ch, 0, Long (ch)-1)
          Sinon
            Retourner  Inconnu (Sous-chaine (ch, 0, Long (ch)-1)
          FinSi
     Fin Si
Fin
```

Travail demandé :

1- Compléter l'entête de la fonction **Inconnu** en complétant la déclaration des paramètres et le type de retour.
2- Dresser le tableau de déclaration des objets locaux de la fonction **Inconnu**.
3- Quel est le résultat retourné par la fonction **Inconnu** pour **ch**="**Bac22G3**".
4- Déduire le rôle de cette fonction ?

Exercice 3 :

Soit l'algorithme de la fonction **Inconnu** suivant :

```
Fonction Inconnu ( a,b : Réel) : ……………………………….
Début
     Si a - b ≥ 0 Alors
        Retourner  a
     Sinon
       Retourner  Inconnu (b,a)
     FinSi
Fin
```

Questions :

En se référant à l'algorithme **Inconnu** et pour chaque des propositions ci-après, remplir la case par la lettre correcte :

Proposition	**1**	**2**	**3**	**4**
Réponse				

1- Le type de la fonction **Inconnu** peut être :
   a) Octet **b)** Réel **c)** Entier long
2- La condition d'arrêt du traitement récursif est :
   a) $a - b \geq 0$ **b)** Retourner a **c)** Retourner Inconnu (b,a)
3- Pour a = 9 et b = 9, le résultat retourné par la fonction **Inconnu** est égal à :
   a) 9 **b)** 12 **c)** 3
4- Le rôle de la fonction **Inconnu** est de :
   a) Calculer le PPCM de a et b **b)** Calculer le PGCD de a et b **c)** Rechercher le maximum de a et b

Exercice 4 : Soit l'algorithme de la fonction **F** suivant:

```
Fonction F (n : entier) : chaîne
Début
    i ← 2
    ch ←""
    Répéter
        Si  n mod i=0  Alors
            Ch1← Convch(i)
            Ch ←ch+ch1+ "*"
            n ←n div i
        Sinon
            i←i +1
        Finsi
    Jusqu'à (n =1)
    ch← effacer ( ch , long ( ch )-1 , long ( ch ) )
    Retourner ch
Fin
```

Questions:

1- Quel est le résultat retourné par la fonction **F** pour **n=30** et pour **n=17**
2- Déduire le rôle de cette fonction ?
3- Proposer un algorithme récursif de la fonction **F**.

Itératif vs récursif :

Ecrire l'algorithme des modules récursifs nommés :

1) **Factorielle** permettant de calculer le factoriel d'un entier *n≥0*, avec n !=n*(n-1)*(n-2)*………..*3*2*1
2) **Palindrome** permettant de vérifier si une chaine donnée non vide est palindrome ou non. *Exemple* : radar, aziza…
3) **Premier** permettant de vérifier si un entier *n* positif est premier ou non.
4) **PGCD** permettant de déterminer le pgcd de deux entiers naturels *a* et *b* par la méthode d'Euclide et la différence.
5) **Occurrence** permettant de déterminer le nombre d'occurrences d'un caractère *Car* dans une chaîne *ch*.
6) **Remplissage_T** permettant de remplir un tableau *T* par *N* entiers.
7) **Affichage_T** permettant d'afficher un tableau *T* par *N* entiers.
8) **Maximum_T** permettant de déterminer la plus grande valeur dans un tableau *T* de *N* entiers.
9) **Minimum_T** permettant de déterminer la valeur minimale dans le tableau *T* de *N* entiers.
10) **Somme_T** permettant de déterminer la somme des éléments d'un tableau T de *N* entiers.
11) **RechercheSeq** permettant de vérifier l'existence d'un entier **x** dans un tableau *T*.
12) **Dichotomique** permettant de vérifier l'existence d'un entier *x* dans un tableau *T* trié dans l'ordre croissant contenant *N* entiers, en utilisant la technique de la recherche dichotomique :

Etape 1 : On compare **x** avec **T[mil]** sachant que **mil = (deb+ fin) div 2** (où **deb** et **fin** respectivement l'indice du début et l'indice de la fin du tableau)

- Si x est égal à T[mil] , la recherche est terminée.
- Si x est inférieur à T[mil], on refait la recherche dans la partie gauche du tableau (de l'indice d à mil-1)
- Si x est supérieur à T[mil], on refait la recherche dans la partie droite du tableau (de l'indice mil+1 à f)

Etape 2 : on refait l'étape 1 pour la partie sélectionnée du tableau jusqu'à trouver l'élément rechercher ou il n'existe pas (d >f).

Exemple : Pour **x=10**, **N=8** et le tableau **T** suivant :

-5	-2	0	4	6	10	15	23
0	1	2	3	4	5	6	7

- On a : d = 0 et f = 7 donc mil = (0+7) div 2 = 3
  T[mil] = T[3] =4 ≠ x
  x est supérieur à T[mil], on refait la recherche dans la partie droite du tableau de l'indice **mil+1** (3+1=**4**) à l'indice **7**
- Pour la partie sélectionnée du tableau d = 4 et f =7 donc :
  Mil = (4+7) div 2 = 5 et T[mil] = T[5] =10 = x, donc la recherche est terminée et **x** se trouve dans le tableau.

13) **Remplissage_M** permettant de remplir une matrice *M* par *N* **M* entiers.
14) **Affichage_M** permettant d'afficher une matrice *M* par *N* **M* entiers.