Interrogation d'ITC n°2

Semaine du 02/12/2024

Durée : 30 minutes

Nom :		**Prénom :**	

Consignes

- Les codes doivent être présentés en mentionnant explicitement l'indentation, au moyen par exemple de barres verticales sur la gauche.
- Pour qu'un code soit compréhensible, il convient de choisir des noms de variables les plus explicites possibles.
- La performance du code proposé à chaque question entrera dans l'évaluation, de-même que l'utilisation de boucles appropriées.
- Vous pouvez bien sûr utiliser une feuille de brouillon en parallèle.

/20

Exercice 1 **Chaîne miroir** Étant donnée une chaîne de caractère, on appelle *chaîne miroir* la chaîne obtenue en lisant la chaîne de caractère de droite à gauche. Ainsi la chaîne miroir de "MPSI" sera "ISPM".

1. /1.5 Compléter la fonction ci-dessous afin qu'elle renvoie la chaîne miroir.

```
def Miroir(S:str)->str:
    r = ________
    for k in range(________):
        r = ________
    return r
```

2. /3 Écrire une fonction récursive `MiroirRec(S)` la chaîne miroir de `S`.

Exercice 2 **Fonction mystère**

```
def F(n:int, m:int)->int:
    if n+1 == m:
        return m
    else:
        c = (n+m)//2
        return F(n,c)*F(c,m)
```

1. /1 Que renvoie `F(4, 5)` ? *on expliquera le résultat*

2. /1 Que renvoie `F(3, 6)` ? *on expliquera le résultat*

3. 1 Soient $(m, n) \in \mathbb{N}^2$ tels que $m > n$. Conjecturer ce que renvoie `F(n, m)`. Comment calculer $n!$ à l'aide de `F` ?

4. 2 Écrire un programme itératif `F_it(n, m)` qui renvoie le même entier renvoyé lors de l'appel `F(n, m)`.

Exercice 3 **Décomposition en base 2** On rappelle que :

- tout entier n se *décompose en base 2* de la manière suivante :
  $$n = \sum_{k=0}^{m} a_k 2^{m-k} = a_0 2^m + \ldots + a_1.2 + a_m,$$
  avec $a_k \in \{0, 1\}$ pour tout $k \in [\![0\,,\,m]\!]$, $m \in \mathbb{N}$.
- On lui associe alors la liste $\mathrm{L} = [a_0, a_1, \ldots, a_m]$.

1. 2 Écrire une fonction itérative `conv(L:list)->int` qui renvoie l'entier dont la décomposition en base deux est L.

2. 3 Écrire une fonction récursive `convRec(L:list)->int` qui calcule le même entier que précédemment, mais de manière récursive. *Cette fonction s'appuiera sur l'algorithme d'HÖRNER, c'est-à-dire sur la formule :*

$$a_0 2^m + \ldots + a_1.2 + a_0 = ((((a_0 \times 2 + \ldots) a_{m-2}) \times 2 + a_{m-1}) \times 2 + a_m.$$

Exercice 4 Fonction mystère Soit la fonction `mystere(t, k)` où `L` est une liste d'entiers non vide et `k` vérifiant $0 \leq k < \mathrm{len}(t)$.

```python
def mystere(L:list, k:int)->bool:
    if k == len(L) - 1:
        return True
    elif L[k] > L[k+1]:
        return False
    else:
        return mystere(L, k+1)
```

1. Soit `L = [6, 9, 4, 8, 12]`.

1.1) 1

Que renvoie `mystere(L, 2)` ? *on expliquera le résultat*

1.2) 1

Que renvoie `mystere(L, 0)` ? *on expliquera le résultat*

2. 1.5

Conjecturer ce que renvoie `mystere` de manière générale.

3. 2

Écrire un programme itératif `mystere_it(L, k)` qui renvoie le même booléen que `mystere`.

SOLUTIONS DES EXERCICES

Solution 1

1. Compléter la fonction ci-dessous afin qu'elle renvoie la chaîne miroir.

```python
def Miroir(S:str)->str:
    r = ""
    for k in range(len(S)):
        r = S[k] + r
    return r
```

```
>>> Miroir("MPSI")
'ISPM'
```

2. Écrire une fonction récursive `MiroirRec(S)` la chaîne miroir de `S`.

```python
def MiroirRec(S:str)->str:
    if len(S) == 0:
        return ""
    else:
        return MiroirRec(S[1:]) + S[0]
```

```
>>> MiroirRec("MPSI")
'ISPM'
```

Solution 2

1. Comme $4+1=5$, on est dans un cas terminal et la fonction renvoie 5.
2. - Nous ne sommes pas dans un cas terminal, on calcule $c = (3+6)//2 = 9//2 = 4$, puis on renvoie `F(3, 4) * F(4, 6)`,
   - ◇ l'appel `F(3, 4)` est terminal, il renvoie 4,
     ◇ l'appel `F(4, 6)` n'est pas terminal, on calcule $c = (4+6)//2 = 5$ donc renvoie `F(4, 5) * F(5, 6)`.
       — ces deux appels sont des cas terminaux, cela renvoie `5*6`.

   Après dépilement : `F(3, 6)` renvoie $4\times5\times6 = \boxed{120}$.
3. De façon générale, on conjecture que `F(n, m)` $= \boxed{\prod_{k=n+1}^{m} k}$. Qour avoir $n!$, on peut exécuter `F(0, n)`

```
>>> F(0, 3)
6
>>> F(0, 4)
24
```

4.

```python
def F_it(n, m):
    P = 1
    for k in range(n+1, m+1):
        P *= k
    return P
```

```
>>> F(4, 5)
5
>>> F_it(4, 5)
5
>>> F(3, 6)
120
>>> F_it(3, 6)
120
```

Solution 3

1.

```python
def conv(L:list)->int:
    n = 0
    m = len(L)-1
    for k in range(len(L)):
        n += L[k]*2**(m-k)
    return n
```

```
>>> conv([1, 2, 3])
11
>>> 1*(2**2)+2*(2**1)+3*(2**0)
11
```

2.

```python
def convRec(L:list)->int:
    if len(L) == 0:
        return 0
    else:
        return L[-1] + 2*convRec(L[:-1])
```

```
>>> convRec([1, 2, 3])
11
>>> 1*(2**2)+2*(2**1)+3*(2**0)
11
```

Solution 4

1. Soit `L = [6, 9, 4, 8, 12]`.

1.1) `mystere(L, 2)`, comme $4 \leqslant 8$, appelle `mystere(L, 3)`,

- `mystere(L, 3)`, puisque $8 \leqslant 12$, appelle `mystere(L, 4)`,
  - `mystere(L, 4)` renvoie **`True`** puisque $4 =$ `len(L)-1`.

1.2) `mystere(L, 0)`, comme $6 \leqslant 9$, appelle `mystere(L, 1)`,

- `mystere(L, 1)`, comme $9 > 4$, renvoie **`False`**.

2. La fonction `mystere(L, k)` semble donc renvoyer `True` si `L[k:]` est croissante et `False` sinon.

3.

```python
def mystere_it(L:list, k:int)->bool:
    n = len(L)
    croi = True
    i = k
    while i < len(L)-1 and croi:
        if L[i] > L[i+1]:
            croi = False
        i += 1
    return croi
```

```python
>>> L = [6, 9, 4, 8, 12]
>>> mystere(L, 2)
True
>>> mystere(L, 0)
False
>>> mystere_it(L, 2)
True
>>> mystere_it(L, 0)
False
```