TPC² — T.C.D'I.

LISTES, LISTES DE LISTES

§1. Constructions

20°

Exercice 1 — En partant à chaque fois d'une liste vide `L = []` et en utilisant uniquement l'instruction `L.append(...)`, écrire les programmes qui construisent les listes suivantes.

1) [1, 2, 3, ..., n],

2) [0, 1, 2, ..., $n-1$],

3) [0, 1, 0, 1, ...] de longueur n,

4) [0, 1, 0, −1, 0, 1, ...] de longueur n,

5) [n, $n-1$, ..., 2, 1],

6) [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, ...] de longueur n,

7) [0, 1, 4, 9, 16, 25, ..., n^2].

30°

Exercice 2

1) Rappeler, si `A` et `B` sont des listes, la signification de `A + B` et de `n * A`.

2) En utilisant uniquement ces deux opérations, écrire les programmes qui fabriquent les listes suivantes : [0, 1, 0, 1, ...] de longueur $2n$, [1, 1, 0, 0, 1, 0, ...] jusqu'au terme d'indice n^2, le terme d'indice k valant 1 si et seulement si k est un carré, et enfin [1, 2, 2, 3, 3, 3, 4, ..., n, ..., n].

20°

Exercice 3 — On donne ce programme, ne renvoyant rien et censé modifier la liste `L`.

```
1 def Vider(L) :
2     L = []
```

1) Tester et commenter.

2) En utilisant la fonction `L.pop()`, écrire une version correcte de ce programme.

50°

Exercice 4

1) Soient a et b deux nombres. Écrire un programme qui renvoie la liste [a, b, a, b, a, b, ...] de longueur n (qui peut donc se finir par a ou par b).

2) Idem pour [a, b, c, a, b, c, a, ...] pour trois nombres a, b et c.

3) Maintenant avec r objets quelconques x_0, x_1, x_2, ..., x_{r-1}.

60°

Exercice 5 — Écrire les programmes qui construisent les listes

1) [1, 2, 3, ..., n, $n-1$, ..., 1],

2) [1, 2, 1, 2, 3, 2, 1, 2, 3, 4, 3, ...] de longueur n,

3) `[0, 1, ..., m, m-1, ..., 0, -1, ..., -m, -m+1, ..., 0, 1, ..., m, m-1, ...]` de longueur n.

20°

Exercice 6

1) Quelle est la différence entre `L_ = L` et `L_ = list(L)` ?

2) Programmer la fonction `SousListe(L, a, b)` qui étant donnée `L = [`x_0`, `x_1`, ...]` construit et renvoie la nouvelle liste `[`x_a`, `x_{a+1}`, ..., `x_{b-1}`]`. C'est la fonction d'extraction prédéfinie `L[a : b]`, qu'on n'utilisera donc pas !

3) Quel est le coût du programme précédent ?

40°

Exercice 7

1) Programmer la fonction `Insérer(L, i, x)` qui étant donnée une liste `L = [`x_0`, `x_1`, `x_2`, ...]` la modifie en `[`x_0`, ..., `x_{i-1}`, `x`, `x_i`, ...]`. Quel est son coût (remarque : c'est la fonction `L.insert(i, x)` prédéfinie) ?

2) Programmer la fonction `Enlever(L, i)` qui étant donnée une liste `L = [`x_0`, `x_1`, `x_2`, ...]` la modifie en `[`x_0`, ..., `x_{i-1}`, `x_{i+1}`, ...]`. Quel est son coût (remarque : c'est la fonction `L.pop(i)` prédéfinie) ?

70°

Exercice 8

1) Écrire un programme `Couples(L)` qui étant donnée une liste `L = [`x_0`, `x_1`, ...]` construit et renvoie la liste de tous les couples `(`x_i`, `x_j`)`.

2) Écrire de même le programme `Triplets(L)`.

3) Plus généralement, écrire un programme `Uplets(L, k)` qui construit tous les k-uplets à composantes dans `L`.

§2. Parcours

20°

Exercice 9 — Écrire un programme `EstDedans(L, x)` qui teste si x est un élément de `L` et renvoie un booléen pour le dire.

```
>>> EstDedans([1, 2, 3], "1")
False
```

60°

Exercice 10 — Écrire les programmes qui étant données deux listes `A` et `B` :

1) détermine si `A` est incluse dans `B` (avec multiplicité),

2) calcule leur réunion (avec multiplicité),

3) calcule leur intersection (avec multiplicité).

Ainsi par exemple l'intersection de `[1, 2, 1, 2]` et `[2, 1, 2]` sera `[1, 2, 2]` (peu importe l'ordre).

20°

Exercice 11 — Écrire les programmes

1) `Somme(L)` et `Produit(L)`,

2) `Maximum(L)`,

3) `IndiceMaximum(L)` renvoyant un indice i tel que `L[`i`]` est maximal.

30°

Exercice 12

1) Écrire le programme `Renversement(L)` qui construit et renvoie la liste-miroir de `L`.

2) Écrire aussi `Renverser(L)` qui modifie `L` en son miroir et ne renvoie rien.

3) Enffin, écrire `EstPalindromique(L)` qui teste si une liste est un palindrôme.

40°

Exercice 13 — Écrire un programme `PPCM(L)` qui calcule et renvoie le PPCM d'une liste d'entiers.

```
>>> PPCM([12, 15, 20])
60
```

40°

Exercice 14

1) Écrire le programme `Occurrences(L, x)` qui renvoie le nombre d'occurrences de x dans la liste `L`.
2) En déduire le programme `Mode(L)` qui détermine l'élément ayant le plus grand nombre d'occurrences dans `L`.
3) Étudier le coût des programmes précédents, en fonction de la longueur n de la liste.
4) Reprendre les questions précédentes dans le cas où la liste est triée.

30°

Exercice 15

1) Écrire un programme `EstCroissante(L)` qui teste si une liste est rangée dans l'ordre croissant (au sens large).
2) De même écrire un programme `EstMonotone(L)` (toujours au sens large).

40°

Exercice 16

1) On donne une liste `[`x_0`, ...]` et pour chaque indice i on pose $m_i = \max(x_0, x_1, \dots, x_i)$. Écrire le programme qui construit la liste `[`m_0`, `m_1`, ...]`.
2) Écrire un programme `SommesPartielles(L)` qui étant donnée une liste `[`x_0`, `x_1`, ...]` construit et renvoie la liste `[`x_0`, `x_0+x_1`, `$x_0+x_1+x_2$`, ...]`. Quel est son coût ?
3) Écrire un programme `Moyennes(L)` qui étant donnée une liste `[`x_0`, `x_1`, ...]` construit et renvoie la liste `[`σ_0`, `σ_1`, ...]` où
$$\sigma_n = \frac{x_0 + x_1 + \cdots + x_n}{n+1}.$$
On cherchera à être aussi efficace que possible.

60°

Exercice 17 — Dans chaque cas suivant, on demande d'écrire un programme qui prend en entrée une liste `[`x_0`, `x_1`, ..., `x_{r-1}`]` et qui la modifie en la liste indiquée. Ce programme ne doit rien renvoyer :

1) `[`x_0`, `x_0`, `x_1`, `x_1`, `x_2`, `x_2`, ...]`,
2) `[`x_0`, ..., `x_{r-1}`, `x_0`, ..., `x_{r-1}`]`,
3) `[`x_0`, `x_1`, `x_1`, `x_2`, `x_2`, `x_2`, ...]` avec $k+1$ fois le terme d'indice k,
4) `[`x_1`, `x_0`, `x_3`, `x_2`, `x_5`, `x_4`, ...]` en supposant la liste de longueur paire,
5) `[`x_0`, `x_0+x_1`, `x_1+x_2`, ..., `$x_{r-2}+x_{r-1}$`, `x_{r-1}`]`.

§3. Cribles

20°

Exercice 18 — On considère une liste `B` de booléens. Écrire un programme `LesVrais(B)` qui renvoie la liste des indices x (dans l'ordre croissant) tels que `B[`x`]` est vrai.

```
>>> LesVrais([True, False, False, True, False, False, True, True, False])
[0, 3, 6, 7]
```

40°

Exercice 19 — **Puissances.** Écrire un programme `PetitesPuissances(M)` qui renvoie la liste des entiers inférieurs ou égaux à M et de la forme a^b avec $a, b \geqslant 2$. On utilise une liste `B`, de longueur $M+1$ et initialisée avec des `False`, et on modifiera les cases pour que `B[`x`]` soit vrai si et seulement si x est une puissance.

40°

Exercice 20 — Écrire un programme `PetitsPremiers(M)` qui renvoie la liste des entiers premiers inférieurs ou égaux à M. On utilisera une liste `B` de booléens, de longueur $M+1$ et initialisée à `[False, False, True,`

`True, True, ...]` puis que des « `True` » jusqu'à la fin, et on modifiera en `False` toutes les cases dont l'indice est un nombre de la forme ab avec $a, b \geqslant 2$. C'est le *crible d'Ératosthène*.

§4. Listes de listes

30°

Exercice 21 — Coefficients binomiaux. On rappelle que le coefficient binomial « k parmi n » est défini par

$$\binom{n}{k} = \frac{n \times (n-1) \times \cdots \times (n-k+1)}{k \times (k-1) \times \cdots \times 1}.$$

1) Écrire un programme `Binomiaux(`n`)` qui construit et renvoie la liste `[1,` n`,` $(n-1)n/2$`, ..., 1]` correspondant à la ligne de rang n dans le triangle de Pascal.

2) Écrire le programme `Pascal(`$n_{\max}$`)` qui renvoie la liste de listes `T` telle que `T[`n`][`k`]` soit le coefficient binomial $\binom{n}{k}$. On ne fera apparaître que les coefficients non nuls.

70°

Exercice 22 — Écrire les programmes qui étant donnée une liste de listes d'entiers renvoient :

1) le plus grand élément,

2) le nombre total d'éléments,

3) la liste la plus longue,

4) la longueur de la liste la plus longue,

5) la liste dont la somme est maximale,

6) le minimum des maximums.

40°

Exercice 23

1) Écrire un programme qui crée un tableau à deux dimensions, sous la forme d'une liste de listes, de taille $m \times n$, dont toutes les cases sont initialisées à la valeur x.

2) Idem pour un tableau à trois dimensions, sous la forme d'une liste de listes de listes.

70°

Exercice 24 — Écrire les programmes qui construisent les listes de listes suivantes.

1) `[[0, 1, 0, 1, ...],`
`[1, 0, 1, 0, ...],`
`...]` le « damier » de taille $m \times n$,

2) `[[1, 2, 3, ..., `n`],`
`[`$n+1$`, ..., `$2n$`],`
`...`
`[..., `$mn-1$`, `mn`]]` de taille $m \times n$,

3) `[[1, 2, 3, ..., `n`],`
`[`$4n-4$`, `$4n-3$`, ..., `$n+1$`],`
`[`$4n-5$`, ..., `$n+2$`],`
`...`
`[`$3n-2$`, ..., `$2n$`, `$2n-1$`]]` de taille n^2, obtenu en enroulant « en spirale » les nombres de 1 à n^2.

50°

Exercice 25 — On souhaite écrire un programme qui étant donnée une liste représentant un ensemble, construit et renvoie la liste de ses parties. Par exemple pour `L = [1, 2, 3]`, on souhaite obtenir `P = [[], [1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3]]`, éventuellement dans un ordre différent.

1) On suppose construite la liste `P` de l'exemple ci-dessus. On considère la liste `L1 = L + [4]`. Expliquer comment obtenir la liste correspondante `P1` facilement à partir de `P`.

2) En déduire le programme demandé.