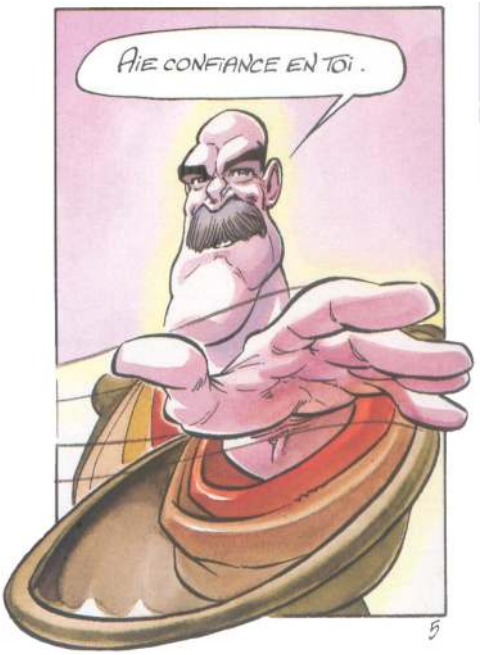
Algorithmique
Contrôle n° 2 (C2)

Info-sup S2#
Epita

30 octobre 2018 - 14 : 30

Consignes (à lire) :

- ☐ Vous devez répondre sur **les feuilles de réponses prévues à cet effet**.
  - — Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  - — Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
  - — Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
  - — Aucune réponse au crayon de papier ne sera corrigée.
- ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- ☐ **Le code :**
  - — Tout code doit être écrit dans le langage `Python` (pas de C, Caml, Algo ou autre).
  - — **Tout code `Python` non indenté ne sera pas corrigé.**
  - — Tout ce dont vous avez besoin (fonctions, méthodes) est indiqué en **annexe** !
- ☐ Durée : 2h00



1 Des expressions et des arbres

Rappel

On peut représenter une expression par un arbre : les nœuds internes contiennent les opérateurs, les opérandes, elles, sont contenues dans les nœuds externes. Les expressions (et donc les arbres) dont il est question dans les 3 prochains exercices sont des expressions arithmétiques **non vides** utilisant **uniquement les opérateurs binaires** + - * et /.

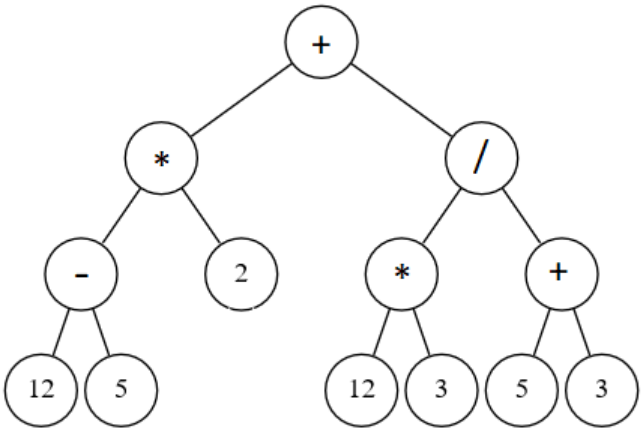


FIGURE 1 – Arbre de l'expression $(12-5)*2+(12*3)/(5+3)$

Exercice 1 (Dessine moi – *4 points*)

Les arbres binaires B_1, B_2 et B_3 représentent des expressions.

Lors du parcours profondeur main gauche

- les valeurs de B_1 sont rencontrées en préfixe dans cet ordre : + - / 8 2 5 + 4 * 2 3
- les valeurs de B_2 sont rencontrées en suffixe dans cet ordre : 8 20 - 2 2 * 2 + /
- les valeurs de B_3 sont rencontrées en infixe dans cet ordre : 8 + 7 / 5 - 4 * 2
- la valeur de l'expression représentée par B_3 est -5.

Dessiner les arbres B_1, B_2 et B_3 (deux solutions pour B_3, une seule demandée) et donner les valeurs des expressions représentées par B_1 et B_2.

Exercice 2 (Compte moi – *3 points*)

Écrire la fonction `nodes` qui retourne le nombre d'opérateurs, ainsi que le nombre d'opérandes d'une expression représentée par un arbre binaire non vide.

Exemple d'application sur l'arbre de la figure 1 :

```
>>> nodes(B)
(6, 7)
```

Exercice 3 (Affiche moi – *2 points*)

Écrire la fonction `exp2str` qui **retourne une chaîne** contenant l'expression complètement parenthésée représentée par un arbre binaire non vide (dont les clés sont des chaînes).

Exemple d'application sur l'arbre de la figure 1 :

```
>>> exp2str(B)
'(((12-5)*2)+((12*3)/(5+3)))'
```

2 Des matrices

Exercice 4 (Symétrique - *4 points*)

La matrice transposée d'une matrice est la matrice A^T, obtenue en échangeant les lignes et les colonnes de A.

$$A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} \text{ alors } A^\mathsf{T} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

Une matrice *symétrique* est une matrice carrée (de taille $n \times n$) qui est égale à sa propre transposée.

Exemples :

Matrice symétrique :

	0	1	2
0	**2**	3	0
1	3	**10**	4
2	0	4	**1**

Matrice non symétrique :

	0	1	2
0	**2**	3	0
1	4	**10**	4
2	0	3	**1**

Écrire la fonction `symmetric` qui teste si une matrice carrée non vide est symétrique.

Exercice 5 (Minimax – *4 points*)

Écrire une fonction qui cherche la valeur minimale parmi les maximums de chaque ligne d'une matrice d'entiers non vide.

Exemples d'application sur les matrices `A` et `B` de l'exercice suivant : :

```
>>> minimax(A)
3
>>> minimax(B)
5
```

Exercice 6 (Mystery – *4 points*)

Soit la fonction suivante `what` et les deux matrices `A` et B suivantes :

```
def what(M):
  (l, c) = (len(M), len(M[0]))

  for j in range(1, c):
      M[0][j] += M[0][j-1]

  for i in range(1, l):
      M[i][0] += M[i-1][0]

  for i in range(1, l):
      for j in range(1, c):
          M[i][j] += min(M[i-1][j], M[i][j-1])

  return M[l-1][c-1]
```

Matrice `A` :

	0	1	2
0	2	3	0
1	2	10	0
2	3	0	1

Matrice `B` :

	0	1	2
0	1	4	5
1	2	3	8
2	5	2	3

1. Donner les résultats de l'application de la fonction aux matrices `A` et `B`. Pour chaque appel :
   — Quelle est la valeur retournée ?
   — Que contient la matrice après exécution ?

2. Soit une matrice `M` non vide de taille $n \times n$: Quel est le nombre exact d'additions effectuées par `what(M)` ?

Annexes

Les arbres binaires

Les arbres binaires manipulés ici sont les mêmes qu'en td.

— `None` est l'arbre vide.

— L'arbre non vide est un objet de la class `BinTree` avec 3 attributs : `key`, `left`, `right`.

```
1  class BinTree:
2      def __init__(self, key, left, right):
3          self.key = key
4          self.left = left
5          self.right = right
```

Fonctions et méthodes autorisées

Sur les listes :

— `len`

Autres :

— `range`

— `abs`

— `min` et `max`, mais uniquement avec deux valeurs entières !

Vos fonctions

Vous pouvez également écrire vos propres fonctions, dans ce cas elles doivent être documentées (on doit savoir ce qu'elles font).

Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.