PA&R TP 2 et 3 **Expressions dans des arbres avec des piles !**

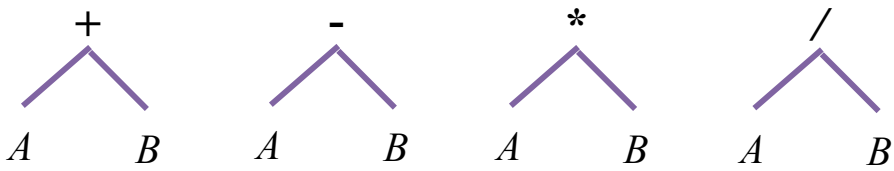
On va considérer des expressions arithmétiques construites à partir de nombres, variables et opérateurs. Dans cet exercice, nous choisissons de représenter ces expressions par des arbres binaires (c'est à dire dont chaque nœud a au plus deux fils).

Plus précisément, les expressions étudiées pourront être :

- un **nombre entier** *r*. Dans ce cas, l'expression sera représentée par un arbre réduit à un seul nœud contenant *r*,
- une **variable**, pour simplifier on se contentera de 3 variables possibles : *X*, *Y* ou Z. Dans ce cas, l'expression sera représentée par un arbre réduit à un seul nœud contenant cette variable,

ou bien, si *u* et *v* sont des expressions représentées par les arbres *A* et *B* :

- la *somme*, la *différence*, le *produit* ou le *rapport* de *u* et *v*. Dans ces cas, les expressions obtenues seront respectivement représentées par les arbres :



1. Définir une classe arbre et une structure noeud permettant de coder en C++ de tels arbres.

2. La notation polonaise inverse

   La notation polonaise inverse permet de noter les formules arithmétiques sans utiliser de parenthèses (http://fr.wikipedia.org/wiki/Notation_polonaise_inverse). Pour cela les expressions sont écrites en plaçant les opérandes avant les opérateurs.

   Par exemple 2 + 3 s'écrit 2 3 +

   et : (((2 + 3 × 4) + 5 × 6 × 7+ 8 × (9 + 10) × 11) + 12) × 13

   s'écrit : 2 3 4 * + 5 6 7 * * + 8 9 10 + * 11 * + 12 + 13 *

   Une expression étant saisie dans une chaîne de caractères sous forme polonaise inverse, écrire une fonction membre de arbre permettant de stocker cette expression dans un arbre.

   Indication :

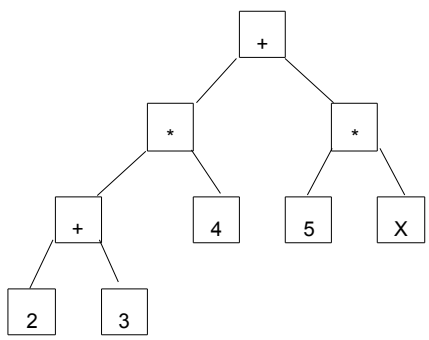
   Une expression sous forme polonaise inverse présentant les opérandes avant les opérateurs, et les arbres étant construit avec les opérateurs en pères des opérandes, la construction de l'arbre nécessite l'utilisation d'une **pile** pour remettre tout ça dans l'ordre.

   Les opérateurs, valeurs et variables seront séparés par des espaces.

3. Écrire la fonction membre (de la classe arbre) afficher qui affiche l'expression codée par un arbre. Cet affichage devra limiter l'affichage des parenthèses au minimum :

   l'expression saisie : 2 3 + 4 * 5 X * +

   sera affichée : ((2 + 3) * 4) + (5 * X)

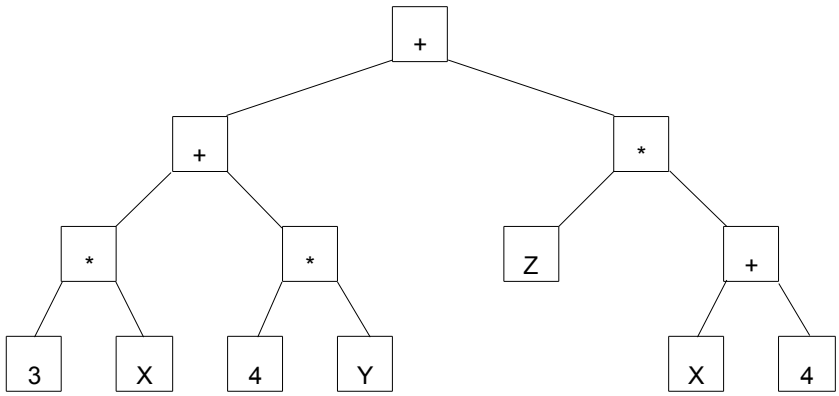


4. Écrire la fonction membre calculer qui prend en argument les valeurs des 3 variables (dans l'ordre X, Y et Z) et qui calcule la valeur d'une expression codée dans un arbre.

   Soit l'expression saisie :

   3 X * 4 Y * + Z X 4 + * +

   codée dans l'arbre A suivant :



   L'appel de A.calculer(8, 11, 22) renverra : 332, le calcul étant :

   ((3 * 8) + (4 * 11)) + (22 * (8 + 4))

   = (24 + 44) + (22 * 12)

   = 68 + 264

   = 332

5. Écrire la fonction membre simplifier qui simplifie au maximum l'expression représentée par un arbre.

   À partir de l'arbre de la question 3 la fonction produira l'arbre suivant :