Exercice 1

CG-USER(1): 23 23 CG-USER(2): (quote 23) 23 CG-USER(3): '23 23	
CG-USER(4):(set x 23) Error: Attempt to take the value of the unbound variable `X'. [condition type: UNBOUND-VARIABLE]	Erreur due à l'absente de (quote), qui aurait empêcher l'évaluation de x.
CG-USER(5): (setq x 32) 32 CG-USER(6): (list x x 32) (32 32 32)	
CG-USER(8): (cadr (list x x 32)) 32	Cadr() = car(cdr())
CG-USER(9): (setq x 'y) Y CG-USER(10): (setq xx 5) 5 CG-USER(11): (setq y '(+ xx xx 32)) (+ XX XX 32) CG-USER(12): x Y CG-USER(13): (eval x) (+ XX XX 32) CG-USER(14): (eval Y) 42 CG-USER(15): (cadr Y) XX	On voit bien l'utilité de (eval).
CG-USER(16): (eval (list '+ (eval y) (cadr y))) 47	La liste est (+ 42 xx), son évaluation est donc 42+5=47.
CG-USER(17): (setq z (+ (if x 2 0) (caddr y) (* y y))) Error: `(+ XX XX 32)' is not of the expected type `NUMBER' [condition type: TYPE-ERROR]	Erreur due à (* y y), alors que y est une liste. L'argument doit être un nombre pour (*).
CG-USER(18): (setq y (list (setq z "Albert") x y)) ("Albert" Y (+ XX XX 32)) CG-USER(19): z "Albert" CG-USER(20): y ("Albert" Y (+ XX XX 32))	
CG-USER(21): (setq x (* x x)) Error: `Y' is not of the expected type `NUMBER' [condition type: TYPE-ERROR]	Même erreur que précédemment, l'argument de (*) doit être un nombre, ici x s'évalue à un autre symbole. N.B. : même en faisant (eval x), ça ne marcherait pas puisque (eval x) = y, une liste.
CG-USER(22): x Y	

Exercice 2

Etude des arbres => Représentation sous forme d'une liste. 3 formes :

- Préfixe (racine gauche droite)
- Postfixe (gauche droite racine)
- Infixe (gauche racine droite)

Liste infixée ((x + 2) / (x – 3)) devient la liste préfixée (/ (+ x 2) (- x 3)).
Fonction LIST permettant de faire cette transformation quelle que soit la liste :

CG-USER(19): (defun inf2pre(L) (list (if (listp (cadr L)) (inf2pre (cadr L)) (cadr L)) (if (listp (car L)) (inf2pre (car L)) (car L)) (if (listp (caddr L)) (inf2pre (caddr L)) (caddr L))))	Définition de la fonction. La fonction consiste à créer une nouvelle liste de 3 éléments. 1° élément. Il s'agit du 2° élément de la liste infixée, donc on utilise (cadr). 2° élément. Il s'agit du 1° élément de la liste infixée, donc on utilise (car). 3° élément. Il s'agit du 3° élément de la liste infixée, donc on utilise (caddr). Clôture de la liste. Clôture de la définition de la fonction.

Remarques générales sur cette fonction :

- Que faire si un élément est une liste ? => Il faut également réordonner cette liste. On rappelle donc la fonction. RÉCURSIVITÉ.
- Condition d'arrêt ? => Quand l'élément n'est plus une liste. On utilise donc :
  (if (listp X) (inf2pre X) X)
- $X \in \{(cadr\ L) ; (car\ L) ; (caddr\ L)\}$. Les parenthèses sont TRES importantes, sinon, le if prend comme premier argument car/cadr/caddr et retourne une erreur, de même si l'oubli de parenthèses se fait plus loin.
- Dans le cas d'un arbre, le 2° élément de la liste infixée (qui devient le 1° élément de la liste préfixée) est la racine, donc jamais une liste (il faut bien un sommet à l'arbre). L'utilité de la récursivité pour cet élément reste néanmoins indiscutable, dans la mesure où la fonction est ainsi généralisée pour toutes les listes, et pas seulement celles issues d'un arbre. La fonction marchera aussi bien pour ((x + 2) / (x – 3)) que pour ((x + 2) (x – 4) (x – 3)).

Testons la fonction pour ces deux listes, justement :

CG-USER(33): (setq X '((X + 2) (X - 4) (X - 3))) ((X + 2) (X - 4) (X - 3)) CG-USER(34): (setq y '((x + 2) / (x - 3))) ((X + 2) / (X - 3)) CG-USER(35): (inf2pre x) ((- X 4) (+ X 2) (- X 3)) CG-USER(36): (inf2pre Y) (/ (+ X 2) (- X 3))	Affection à X d'une liste. *Idem* pour Y. Utilisation de la fonction pour X. Tout se déroule comme prévu. De même pour Y.