```
> #Exercice 1: Ecrire une procédure "longueurliste" qui prend une
  liste d'entiers (le dernier élément ne sera pas un entier), qui
  est soit vide, soit dont le dernier élément est F, en argument et
  renvoie la longueur de cette liste, et cela de trois façon
  différentes, dont une en utilisant une boucle "for" et une autre
  avec un "while". Par définition, pour cette éxercice, la longueur
  d'une liste non vide est égale à la longueur de la liste moins 1
  (on ne compte pas le dernier élément F qui marque la fin de la
  liste), et la longueur de la liste vide [] est zéro.
  (Fonction qui peut être utile : nops.)

  #Exercice 2 : Ecrire une procédure OpListe qui prend en argument
  une liste de nombres ou de polynômes, ainsi que soit `+` soit `*`
  (avec les simples quotes) et renvoie la somme (ou le produit) des
  éléments de cette liste. Faire cela de trois façons différentes
  (une version avec un "for", une version avec "while" et une
  version utilisant "convert"). Exemple, OpListe([2,3,2],+) doit
  renvoyer 2+3+2=7, alors que OpListe([1,2,2],*) doit renvoyer 4=1*
  2*2. De même OpListe([x^2+x,x+1],+) renvoie x^2+2*x+1.
  (Fonction utile : nops.)

  #Exercice 3 : Ecrire une procédure qui prend deux entiers a et b
  en arguments, avec 0<=a<b et qui calcule l'inverse de a modulo b,
  lorsque c'est possible, et renoie l'infini (infinity) sinon. Par
  exemple, inverse(3,5) doit renvoyer 2 (car 3*2=6=1 (mod 5)) et
  inverse(2,4) doit renvoyer infinity car 2 n'est pas inversible
  modulo 4.
  (Fonctions utiles : irem, igcd.)

  #Exercice 4 : Ecrire une procédure qui prend une liste de réels et
  renvoie la somme des éléments négatifs de cette liste. Même chose
  en remplaçant la somme par le produit. Vous écrirez les procédures
  sous deux formats : l'un avec for et l'autre avec while.
  Par exemple, somneq([2,-5,3,0,-2]) renvoie -7=-5-2.

  #Exercice 5 : Ecrire une procédure "sfi" ("sfi" signifiant "square
  -free integers") qui prend un entier strictement positif en
  argument et renvoie true si dans la décomposition de cet entier en
  facteurs premiers, chaque facteur premier n'apparaît qu'une fois,
  et false dans le cas contraire. Puis écrire une procédure
  "multisfi" qui prend deux entiers strictement positifs, et renvoie
  le produit des deux entiers à condition que ce produit n'ait pas
```

de facteurs premiers apparaissant plus d'une fois, dans le cas contraire, "multisfi" renvoie 0.
Par exemple, sfi(12) renvoie true car 12=2*3 alors que sfi(24) renvoie false car 24=8*3=2^3*3. De plus, multisfi(3,10) renvoie 30 car 30=2*15=2*3*5. Alors que multisfi(2,12) renvoie 0 car 2*12=24=2^3*3.
(Fonctions utiles : isprime, irem.)