TP3. Un peu d'arithmétique.

Lancer `xmaple`.
Si vous êtes perdu, utiliser l'aide de maple : ? suivi du nom de la commande.

Exercice 1 : Boucle `while..do`. Pour effectuer une boucle un nombre de fois indéterminé, on utilise une boucle `while..do` de la forme

```
while condition do
instruction
end do;
```

qui effectue "instruction" tant que "condition" est remplie. Par exemple, tester

```
x:=1;
while x<20 do
x:=x+1
end do;
```

a) Est-ce une bonne idée d'effectuer le programme suivant ? (Ne pas tester...)

```
x:=1;
while x>0 do
x:=x+1
end do;
```

b) Écrire une procédure `etage` qui à un réel positif y associe le plus grand entier inférieur ou égal à $\sqrt{y}$, c'est-à-dire la partie entière de $\sqrt{y}$. On utilisera `evalf` pour évaluer la racine carrée d'un nombre, et `<=` pour inférieur ou égal.

Exercice 2 : On cherche à écrire une procédure donnant le plus grand commun diviseur (pgcd) de deux entiers par l'algorithme d'Euclide. Décrivons l'algorithme. Soit $a > b$ deux entiers. On commence par poser $a_0 = a$, $a_1 = b$, et on fait la division euclidienne de a_0 par a_1. On note alors a_2 le reste de cette division. Si a_2 est nul, le pgcd cherché est a_1. Si a_2 est non-nul, on remplace a_0 par a_1, a_1 par a_2, et on fait de même la division euclidienne de a_1 par a_2, et ainsi de suite jusqu'à obtenir 0. On obtient ainsi une suite a_n définie par : tant que a_{n+1} est non-nul, a_{n+2} est le reste de la division euclidienne de a_n par a_{n+1}. La suite vaut alors 0 à partir d'un certain rang (pourquoi ?), et le pgcd cherché est alors le terme précédent de la suite.

Exemple : On cherche le pgcd de 1024 et 52. Par l'algorithme d'Euclide, on a :

$$\begin{aligned} 1024 &= 52 \times 19 + 36 \\ 52 &= 36 \times 1 + 16 \\ 36 &= 16 \times 2 + 4, \\ 16 &= 4 \times 4 + 0, \end{aligned}$$

et donc le pgcd cherché est 4.
Les fonctions pré-définies de maple `irem` et `iquo` permettent d'effectuer la division euclidienne de a et b. La fonction `irem` renvoie le reste de la division euclidienne, et la fonction `iquo` le quotient. Exemple :

```
> irem(1024,52);
        36
> iquo(1024,52);
        19
```

a) Écrire l'algorithme d'Euclide sous forme de procédure.

b) Donner le pgcd de 3471 et 1547.

c) L'algorithme marche-t-il si $a < b$? Expliquer pourquoi.

Exercice 3 : Le crible d'Ératosthène est un algorithme permettant de trouver tous les nombres premiers inférieurs à un certain entier N. On rentre les entiers de 2 à N dans une liste, puis on élimine tous les multiples d'un entier : on commence par les multiples de 2, puis les multiples du plus petit entier restant, donc 3, puis 5, et ainsi de suite... Les entiers résultants sont alors les nombres premiers inférieurs à N.

a) On remarque qu'en fait on peut arrêter l'algorithme au plus grand entier inférieur ou égale à $\sqrt{N}$. Expliquer pourquoi.

b) Variables booléennes. Une variable booléenne est une variable à deux états possibles : soit l'état `true` (pour vrai), soit l'état `false` (pour faux). Exemple d'utilisation :

```
> c:=true;
> if c then print(‘ah oui, c’est vrai‘) else print(‘non...‘);
```

On va se servir de variables booléennes pour écrire un algorithme implémentant le crible d'Ératosthène.

Pour cela, on introduit des variables `A[j]`, pour j allant de 2 à N, toutes égales à `true`.

Pour j allant de 2 à la partie entière de $\sqrt{N}$, on fait alors :

si `A[j]` est vrai alors pour tous les multiples k de j compris entre 2 et N, on décide que `A[k]` est faux.

On construit alors la suite des nombres premiers inférieurs à N en ne gardant que les entiers p pour lesquels `A[p]` est vrai.

c) Implémenter le crible d'Ératosthène sous forme de procédure suivant la description ci-dessus.

d) Afficher les nombres premiers inférieurs à 1000. Combien y en a-t-il ?