I. Programme ou algorithme ?

1. Définition

Un algorithme peut se définir comme étant l'ensemble des règles et instructions à suivre, effectivement exécutables, permettant d'obtenir un résultat clairement défini en un nombre fini d'étapes.
Autrement dit, c'est la façon de décrire dans les moindres détails comment procéder pour faire quelque chose.

Il ne faut pas confondre *algorithme* et *programme*.
Un algorithme peut s'exprimer avec une notation indépendante de tout langage de programmation (langage naturel ou *pseudo-code*) et il s'implémente dans un programme qui, lui, est écrit dans un langage particulier compréhensible par une machine. De plus, un algorithme doit toujours donner une réponse après un nombre fini d'opérations, alors que l'exécution d'un programme peut conduire à une boucle infinie et ne jamais s'arrêter.

2. Notion d'instruction élémentaire

Dans une recette de cuisine, on trouve la consigne « rôtir des légumes ». Cela demande en réalité de couper les légumes en petits morceaux, de les mettre dans un plat, de les recouvrir d'huile, puis de mettre le plat au four.

Pour un grand chef, *rôtir des légumes* est une instruction élémentaire. Pour un élève de Terminale S, ça ne l'est probablement pas, et il faut préciser les étapes à l'intérieur de cette consigne. Mais dans les deux cas, pour que les légumes soient finalement rôtis, le grand chef et l'élève de TS auront suivi la même série d'étapes. La différence réside dans l'interprétation automatique qu'en aura faite le grand chef.
De la même manière, il existe plusieurs types de langages de programmation. Dans tous les cas, les étapes les plus élémentaires auxquelles il faut arriver sont celles qui seront comprises par le CPU de la machine.
Un langage de *bas niveau* nécessitera peu d'interprétation, donc il faudra lui fournir des instructions proches des instructions machine.
À l'inverse, un langage de *haut niveau* nécessitera de traduire des instructions très complexes en langage machine.
Cette étape d'interprétation est faite par un *compilateur*. Un compilateur va donc être différent selon le langage utilisé et la machine pour laquelle il exécute la compilation.

Exercice 1 : Observer l'algorithme suivant et préciser ce qu'il fait.

Algorithme en langage naturel

```
début

entier DIM ← 10 ;
tableau d'entiers tab[DIM] = {0,2,9,5,0,8,0,1,4,0};
entier n ;

Pour n allant de 0 à DIM-1 faire
 Si tab[n] égal à 0 faire
  écrire L'élément n+1 du tableau est nul.
 finSi
finPour

fin
```

Programme en langage C

```c
#include <stdio.h>
#define DIM 10
int main(void) {
 /* Entrées */
 int tab[DIM] = {0,2,9,5,0,8,0,1,4,0};
 int n;
 /* Traitements et Sorties*/
 for (n = 0; n < DIM; n++) {
  if (tab[n] == 0) {
   printf("\nL'élément %d du tableau est nul.\n", n+1);
  }
 }
 return 0;
}
```

II. Comment trier les nombres ?

Exercice 2 : Dix élèves se placent sur des chaises alignées dans la salle. Chaque élève assis reçoit une affiche avec un nombre. Un autre élève joue le rôle de chef d'orchestre : son objectif est de placer les élèves assis dans l'ordre croissant des étiquettes. Pour cela, il leur donne des consignes aussi simples et précises que possible.
À un instant donné, un seul élève parmi les dix est autorisé à être debout.
Les élèves spectateurs notent la succession des étapes pour reconstituer l'algorithme mis en œuvre.

1. Le tri par insertion

C'est le tri pratiqué intuitivement pour classer un jeu de cartes : il consiste à ranger toutes les cartes, l'une après l'autre, dans l'ordre de la donne au bon endroit parmi les cartes déjà triées.

Exemple d'algorithme pour un classement de cartes dans l'ordre croissant :

```
Début
Placer la première carte de la donne (dans le tas trié).
Pour chaque carte suivante faire
  Tant que la carte à classer est supérieure à la carte en cours du tas trié :
    Avancer à la carte suivante du tas trié
  FinTant que
FinPour
Fin
```

Remarque : il faudra faire un test pour savoir s'il reste des cartes à classer.

2. Le tri par sélection

C'est également un tri intuitif utilisé pour classer les cartes : il consiste à parcourir l'ensemble des cartes pour trouver la carte suivante à classer, dans l'ordre dans lequel on veut les classer, plutôt que de traiter les cartes dans l'ordre de la donne.

Exemple d'algorithme pour un classement de nombres dans l'ordre croissant :

```
Début
entiers i, j, n, min ;
tableau d'entiers tab[n] ;
Pour i allant de 1 à n – 1 faire
  min ← i
  pour j de i + 1 à n faire
    si tab[j] < tab[min], alors min ← j
  FinPour
  si min ≠ i, alors échanger tab[i] et tab[min]
FinPour
Fin
```

3. Le tri à bulles

Le tri à bulles est un algorithme qui consiste à faire remonter progressivement les plus grands éléments d'un tableau, comme les bulles d'air remontent à la surface d'un liquide.
Ce tri est peu performant et il n'est donc quasiment pas utilisé en pratique.

Exemple d'algorithme pour un classement de nombres dans l'ordre croissant :

```
Début
entiers i, j, n ;
tableau d'entiers tab[n] ;
Pour i allant de n-1 à 1 faire
  Pour j allant de 0 à i-1 faire
    si tab[j+1] < tab[j], alors échanger tab[j] et tab[j+1]
  FinPour
FinPour
Fin
```