Les invariants de boucle:

A priori, les algorithmes de tri par insertion et de tri par sélection "fonctionnent" correctement : ils trient bien le tableau donné en entrée, on dit que ces algorithmes sont corrects. On parle aussi de la "correction d'un algorithme" pour dire qu'un algorithme produit bien le résultat attendu.

Multiplier les exemples qui "fonctionnent" ne veut pas dire que l'algorithme donnera le "bon résultat" dans toutes les circonstances. C'est un peu comme en mathématiques, vérifier qu'une propriété est vraie sur un exemple n'a pas valeur de démonstration. Il se pourrait très bien que dans une situation donnée notre algorithme ne donne pas le résultat attendu.

Il existe un moyen de démontrer (au sens mathématique du terme) la correction d'un algorithme. Si nous arrivons à démontrer la correction de l'algorithme de tri par insertion, nous pourrons alors affirmer que, quel que soit le tableau donné en entrée, nous obtiendrons bien en sortie ce même tableau, mais trié. Pour démontrer la correction de l'algorithme de tri par insertion, nous allons utiliser un "invariant de boucle".

Qu'est-ce qu'un invariant de boucle?

On appelle invariant de boucle une propriété qui est vraie avant et après chaque itération. Prenons tout de suite un exemple avec le tri par insertion (Afin de travailler sans avoir à faire des aller-retour entre les pages, voici, pour mémoire, l'algorithme de tri par insertion):

```
VARIABLE
t : tableau d'entiers
i : nombre entier
j : nombre entier
k : nombre entier
DEBUT
j←2
tant que j<=longueur(t):   //boucle 1
  i←j-1
  k←t[j]
  tant que i>0 et que t[i]>k:   //boucle 2
    t[i+1]←t[i]
    i←i-1
  fin tant que
  t[i+1]←k
  j←j+1
fin tant que
FIN
```

Divisons le tableau t en 2 parties:

- Une partie qui contient les éléments d'index 1 à j-1 (le j correspond à la variable j dans l'algorithme ci-dessus). Nous noterons cette partie (ce tableau) t[1..j-1],
- Une seconde partie qui contient le reste du tableau t (index j à n si le tableau t comporte n élément).

Nous allons travailler sur cet exemple : t = [27, 10, 12, 8, 11].

Au départ nous avons j=2, donc t[1..j-1] = t[1..1] = [27] (le tableau t[1..j-1] contient un unique élément)

À début de la deuxième itération de la "boucle 1", nous avons t = [10, 27, 12, 8, 11] et j=3, d'où t[1..j-1] = t[1..2] = [10, 27]

À début de la troisième itération de la "boucle 1", nous avons t = [10, 12, 27, 8, 11] et j=4, d'où t[1..j-1] = t[1..3] = [10, 12, 27]

À début de la quatrième itération de la "boucle 1", nous avons t = [8, 10, 12, 27, 11] et j=5, d'où t[1..j-1] = t[1..4] = [8, 10, 12, 27]

Que peut-on dire du tableau t[1..j-1] au début de chaque itération?

Réponse : t[1..j-1] est un tableau trié!

Notre invariant de boucle pourrait donc être: "Le tableau t[1..j-1] est trié"

Trouver ce qui pourrait être un invariant de boucle est une chose, encore faut-il ensuite démontrer qu'il est correct (une fois de plus l'étude d'un cas particulier ne vaut pas démonstration). La démonstration doit se faire en 3 étapes:

- INITIALISATION : on doit montrer que l'invariant de boucle est vrai avant la première itération de la boucle
- CONSERVATION : on doit montrer que si l'invariant de boucle est vrai avant une itération de la boucle, il le reste avant l'itération suivante.
- TERMINAISON : une fois la boucle terminée, l'invariant fournit une propriété utile qui aide à montrer la correction de l'algorithme.

Revenons au tri par insertion et à notre invariant de boucle "Le tableau t[1..j-1] est trié":

- INITIALISATION : avant la 1re itération de la boucle 1, le tableau t[1..j-1] contient un seul élément (j=2). Un tableau contenant un seul élément est forcément trié (trivial), notre invariant "le tableau t[1..j-1] est trié" est donc vrai.
- CONSERVATION : nous devons maintenant montrer que l'invariant est conservé au cours d'une itération : si l'invariant est vrai en début d'itération (début de la boucle 1), il reste vrai à la fin de cette même itération. Globalement, que fait la boucle 1 ? Elle permet de déplacer les éléments t[j-1], t[j-2], t[j-3], etc d'une position vers la droite jusqu'à ce que l'on trouve la bonne position pour t[j]. Après ces décalages vers la droite effectués, on insère la valeur t[j] au bon endroit. Au cours d'une itération de la boucle 1, nous passons d'un tableau t[1..j] non trié (le tableau t[1..j-1] est trié, mais l'élément t[j] n'est pas encore à la bonne place, donc t[1..j] n'est pas encore trié), à un tableau trié (décaler vers la droite les éléments t[j-1], t[j-2], t[j-3], etc et placer t[j] à la bonne position). En fin d'itération (juste avant le "$j \leftarrow j + 1$") t[1..j] est trié, juste après le "$j \leftarrow j+1$" le tableau t[1..j] précédemment évoqué devient donc le tableau t[1..j-1]. Conclusion : en toute fin d'itération (après le "$j \leftarrow j + 1$") le tableau t[1..j-1] est trié. Au début d'une itération t[1..j-1] est trié, à la toute fin de cette même itération (après le "$j \leftarrow j + 1$") le tableau t[1..j-1] est trié, nous pouvons donc affirmer que si l'invariant "le tableau t[1..j-1] est trié" est vrai avant une itération de la boucle 1, il le reste avant l'itération suivante.

- TERMINAISON : Que se passe-t-il quand la boucle se termine ? La boucle 1 se termine quand $j > longueur(t)$ (quand $j > n$ pour un tableau comportant n élément). Quand la boucle se termine, on doit avoir $j = n+1$. Si on substitue n+1 par j dans l'invariant de boucle, on obtient le tableau t[1..n]. Quand la boucle se termine, l'affirmation "le tableau t[1..j-1] est trié" devient "le tableau t[1..n] est trié". Or, le tableau t[1..n] correspond au tableau qui est composé des n éléments. Le tableau d'origine est donc maintenant trié.

Cette démonstration nous permet d'affirmer que l'algorithme de tri par insertion est correct.

Exercice: Trouvez un invariant de boucle pour l'algorithme de tri par sélection. Procédez ensuite à la démonstration en 3 étapes afin de démontrer la correction de l'algorithme de tri par sélection.