Devoir surveillé n°6 – NSI
Correction

Exercice 1

1. (a) La boucle la plus interne est répétée `n` fois et contient une somme (`x+j`), ce qui fait donc `n` sommes. Il y a une somme de plus ensuite (`x+i`), ce qui fait donc `n+1`.
   Ces `n+1` sommes sont répétées 4 fois dans la boucle principale (`range(2,6)`, donc valeurs de `i` entre 2 et 5 incluses, ce qui fait 4 valeurs).
   Ainsi, le nombre de sommes est $4 \times (n+1) = 4n+4$.
   (b) L'expression $4n+4$ est affine en `n`, donc l'algorithme a un coût linéaire.
2. (a) Le nombre d'itérations est le nombre de fois qu'il faut ajouter `1/n` depuis `0` (valeur initiale de `x`) pour atteindre `n`.
   Or $n^2 \times \frac{1}{n} = n$, donc il faut n^2 itérations.
   Le coût est donc quadratique.
   (b) Le nombre d'itérations est le nombre de fois qu'il faut multiplier par `2` depuis `1` (valeur initiale de `x`) pour atteindre `n`.
   En fait, la valeur de `x` à la k^e itération est 2^k.
   Le nombre d'itérations nécessaires est donc le plus petit `k` tel que $2^k \geqslant n$
   On a donc `k` de l'ordre de $\log_2(n)$.
   Le coût est donc logarithmique.
   (c) Le nombre d'itérations est le nombre de fois qu'il faut ajouter `2` depuis `0` (valeur initiale de `x`) pour atteindre `n`.
   Or $\frac{n}{2} \times 2 = n$, donc il faut $\frac{n}{2}$ itérations.
   Le coût est donc linéaire.

Exercice 2

1. Voici le tableau complété :

`i` (itération de la boucle)	avant la boucle	1	2	3	4
`i<n`		V	V	V	F
`liste[i]`		2.3	7	6	
`liste[i]>maxi`		F	V	F	
`maxi`	`liste[0]=5`	5	7	7	

2. Ainsi, `fonction([5,2.3,7,6])` retourne `7` (le maximum de la liste, la valeur de `maxi` à la fin de l'exécution de la fonction).
3. La variable `i` est un variant de la boucle `while` : elle augmente de `1` à chaque itération, en commençant à `0`. Elle finira donc par atteindre `n`, ce qui rendra la condition `i<n` fausse et permet de sortir de la boucle.
   Ainsi, la boucle termine.
4. On veut démontrer que la propriété « `maxi` est le maximum de `liste[0:i]` et `i` $\leqslant$ `n` » est un invariant de la boucle. Nommons $\mathcal{P}$ cette propriété.

   Initialisation : Avant la boucle, on a `i=1` et `maxi=liste[0]`.
   Or, `liste[0:i]=liste[0:1]` est la liste qui ne contient que `liste[0]`.
   La variable `maxi` ayant cette valeur, elle est donc bien le maximum de cette liste.
   De plus, comme `n` $\geqslant 1 > 0 =$ `i`, on a bien `i` $\leqslant$ `n`

Hérédité : On suppose qu'avant une itération de la boucle, la propriété $\mathcal{P}$ est vraie.

On note `maxi`$'$ et `i`$'$ les valeurs respectives de `maxi` et `i` à la fin de l'itération.

On doit démontrer que « `maxi`$'$ est le maximum de `liste[0:i']` et `i`$' \leqslant$ `n` ».

Or on sait que `maxi` est le maximum de `liste[0:i]` par hypothèse.

D'après l'algorithme, il y a deux cas :

- Si `liste[i]>maxi`, alors `maxi`$'$`=liste[i]`
- Sinon, `maxi`$'$ reste égale à `maxi`

Dans les deux cas, `maxi`$'$ est alors le maximum de la liste `liste[0:(i+1)]`, car celui-ci est le maximum entre le maximum `maxi` de `liste[0:i]` avec `liste[i]`, ce que détermine la structure conditionnelle de l'algorithme.

Or, à la fin de l'itération, `i`$' =$ `i+1`.

Donc `maxi`$'$ est bien le maximum de `liste[0:i']`.

D'autre part, comme on exécute une itération de la boucle, on a en fait `i<n` (la condition est vraie), donc `i`$' =$ `i+1`$\leqslant$ `n`.

La propriété est donc vérifiée.

Conclusion : La propriété $\mathcal{P}$ est donc bien un invariant de boucle.

5. À la fin de la boucle :
   - La propriété $\mathcal{P}$ est vraie (puisque c'est un invariant), c'est à dire que l'on a : `maxi` est le maximum de `liste[0:i]` et `i`$\leqslant$ `n`.
   - La condition `i<n` est fausse, donc on a `i`$\geqslant$ `n`

   Autrement dit, `i=n` et `maxi` est le maximum de `liste[0:n]` qui est exactement `liste`.

Devoir surveillé n°6 – NSI
Correction

Exercice 1

1. (a) La boucle la plus interne est répétée `3` fois (`range(2,5)`, donc valeurs de `j` entre 2 et 4 incluses, ce qui fait 3 valeurs) et contient une somme (`x+j`), ce qui fait donc 3 sommes.
   Il y a une somme de plus ensuite (`x+i`), ce qui en fait donc 4.
   Ces 4 sommes sont répétées `n` fois dans la boucle principale.
   Ainsi, le nombre de sommes est $4 \times \texttt{n}$.
   (b) L'expression $4 \times \texttt{n}$ est linéaire en `n`, donc l'algorithme a un coût linéaire.
2. (a) Le nombre d'itérations est le nombre de fois qu'il faut multiplier par `2` depuis `1` (valeur initiale de `x`) pour atteindre `n`.
   En fait, la valeur de `x` à la `k`$^{\text{e}}$ itération est $2^{\texttt{k}}$.
   Le nombre d'itérations nécessaires est donc le plus petit `k` tel que $2^{\texttt{k}} \geqslant \texttt{n}$
   On a donc `k` de l'ordre de $\log_2(\texttt{n})$.
   Le coût est donc logarithmique.
   (b) Le nombre d'itérations est le nombre de fois qu'il faut ajouter `1/n` depuis `0` (valeur initiale de `x`) pour atteindre `n`.
   Or $\texttt{n}^2 \times \dfrac{1}{\texttt{n}} = \texttt{n}$, donc il faut $\texttt{n}^2$ itérations.
   Le coût est donc quadratique.
   (c) Le nombre d'itérations est le nombre de fois qu'il faut ajouter `2` depuis `0` (valeur initiale de `x`) pour atteindre `n`.
   Or $\dfrac{\texttt{n}}{2} \times 2 = \texttt{n}$, donc il faut $\dfrac{\texttt{n}}{2}$ itérations.
   Le coût est donc linéaire.

Exercice 2

1. Voici le tableau complété :

étape de la boucle	avant	1	2	3	4	5
`i<n`		V	V	V	V	F
`liste[i]`		5	2.3	7	6	
`res`	0	5	7.3	14.3	20.3	
`i` (fin d'itération)	0	1	2	3	4	

2. Ainsi, `fonction([5,2.3,7,6])` retourne `20.3` (dernière valeur de `res`, qui est la somme des termes de la liste).
3. La variable `i` est un variant de la boucle `while` : elle augmente de `1` à chaque itération, en commençant à `0`. Elle finira donc par atteindre `n`, ce qui rendra la condition `i<n` fausse et permet de sortir de la boucle.
   Ainsi, la boucle termine.
4. On veut démontrer que la propriété « `res = liste[0]+...+liste[i-1]` et $\texttt{i} \leqslant \texttt{n}$ » est un invariant de la boucle. Nommons $\mathcal{P}$ cette propriété.

   Initialisation : Avant la boucle, on a `i=0` et `res=0`.
   Comme `i-1=-1` est négatif, d'après l'énoncé on peut affirmer que la somme :
   `liste[0]+...+liste[i-1]` est nulle, donc `res` a bien la bonne valeur.
   De plus, $\texttt{i} = 0 \leqslant \texttt{n}$ car `n` est un entier naturel (c'est la longueur de la liste).

Hérédité : On suppose qu'avant une itération de la boucle, la propriété $\mathcal{P}$ est vraie.

On note `res`$'$ et `i`$'$ les valeurs respectives de `res` et `i` à la fin de l'itération.

On doit démontrer que « `res`$'$`=liste[0]+...+liste[i`$'$`-1]` et `i`$' \leqslant$ `n` ».

Or, d'après l'algorithme, et du fait que `i`$' =$ `i+1`, donc que `i`$'$`-1=i`,

$$\begin{aligned} \texttt{res}' &= \texttt{res+liste[i]} \\ &= \texttt{liste[0]+...+liste[i-1]+liste[i]} \qquad \text{(par hypothèse)} \\ &= \texttt{liste[0]+...+liste[i}'\texttt{-1]} \end{aligned}$$

D'autre part, comme on exécute une itération de la boucle, on a en fait `i<n` (la condition est vraie), donc `i`$' =$ `i+1`$\leqslant$ `n`.

La propriété est donc vérifiée.

Conclusion : La propriété $\mathcal{P}$ est donc bien un invariant de boucle.

5. À la fin de la boucle :
   - La propriété $\mathcal{P}$ est vraie (puisque c'est un invariant), c'est à dire que l'on a : `res = liste[0]+...+liste[i-1]` et `i`$\leqslant$ `n`.
   - La condition `i<n` est fausse, donc on a `i`$\geqslant$ `n`

   Autrement dit, `i=n` et `res=liste[0]+...+liste[n-1]`, ce qui est exactement la somme des termes de la liste.