❑ **Exercice 1** : *Algorithmes de tri*

1. Tri par sélection
   a) Expliquer en quelques phrases le principe de l'algorithme du tri par sélection

   1 points
   Voirs cours

   b) On applique l'algorithme du tri par sélection à la liste `[11,17,14,42,5,30,19]`. Donner l'évolution du contenu de la liste au cours des étapes du tri.

   2 points
   `[11,17,14,42,5,30,19]`
   `[5,17,14,42,11,30,19]`
   `[5,11,14,42,17,30,19]`
   `[5,11,14,17,42,30,19]`
   `[5,11,14,17,19,30,42]`

2. Tri par insertion
   a) Expliquer en quelques phrases le principe de l'algorithme du tri par insertion

   1 points
   Voir cours

   b) On applique l'algorithme du tri par insertion à la liste `[11,17,14,42,5,30,19]`. Donner l'évolution du contenu de la liste au cours des étapes du tri.

   2 points
   `[11,17,14,42,5,30,19]`
   `[11,14,17,42,5,30,19]`
   `[11,14,17,42,5,30,19]`
   `[5,11,14,17,42,30,19]`
   `[5,11,14,17,30,42,19]`
   `[5,11,14,17,19,30,42]`

   🛟 On pourra donner l'état de la liste après chaque insertion, sans écrire les étapes intermédiaires de ces insertions.

❑ **Exercice 2** : *Programmation en python*

On veut écrire une fonction `trouve_mini` qui renvoie l'indice de la première occurrence du plus petit élément d'une liste. Par exemple, `trouve_mini([13,9,10,7,18])` doit renvoyer 3.

1. Que doit renvoyer `trouve_mini([15,10,17,11,22])` ?

   0.5 points
   `trouve_mini([15,10,17,11,22])` doit renvoyer `1` car le plus petit élément (10) se trouve à l'indice 1

2. Que doit renvoyer `trouve_mini([5,22,5,41,20,5,17])` ?

   0.5 points
   `trouve_mini([5,22,5,41,20,5,17])` doit renvoyer `0` car le plus petit élément (5) apparaît pour la première fois à l'indice 0

3. 🛟 On rappelle que lorsque le minimum apparaît plusieurs fois, on doit renvoyer la première occurence.

4. Compléter le code de la fonction `trouve_mini` ci-dessous. 4 points

```python
def indice_minimum(liste):
indice_minimum = 0
for indice in range(len(liste)):
    if liste[indice]<liste[indice_minimum]:
        indice_minimum = indice
return indice_minimum
```

5. Que renverra `trouve_mini([])` ?

1 points
La boucle n'est jamais parcourue si la liste est vide donc la valeur d'initialisation 0 est renvoyée.

6. Modifier cette fonction afin que `trouve_mini([])` renvoie -1. **1** points

```python
def indice_minimum(liste):
if liste==[]:
    return -1
indice_minimum = 0
for indice in range(len(liste)):
    if liste[indice]<liste[indice_minimum]:
        indice_minimum = indice
return indice_minimum
```

❑ Exercice 3 : *Coût*

1. Expliquer rapidement ce que signifie une complexité linéaire pour un algorithme.

1 points
Cela signifie que lorsque la taille des données est multiplié par un facteur k alors le coût de l'algorithme est multiplié lui aussi par un facteur k

2. Donner au moins un exemple, d'algorithme ayant une complexité linéaire.

1 points
L'algorithme de recherche simple d'un élement dans une liste. On peut aussi citer : recherche du maximum, calcul de la somme, ...

3. Un algorithme de complexité linéaire traite une liste de 25 000 éléments en $0,03s$. Quel est le temps approximatif d'exécution prévisible pour une liste de 150 000 éléments ?

2 points
La taille de la liste a été multiplié par 6, le temps sera donc lui aussi multiplié approximativement par 6, l'exécution devrait donc prendre environ $0,03 \times 6 = 0,18s$

4. On rappelle que : « l'algorithme du tri par sélection à une *complexité quadratique* », expliquer en quelques lignes ce que signifie cette phrase.

1 points
Cela signifie que lors la taille des données est multipliée par un facteur k alors le coût de l'algorithme est multiplié par un facteur k^2.

5. On a écrit un programme permettant de trier une liste à l'aide du tri par sélection. Si ce programme traite une liste de 5 000 élément en 0,75 secondes, quel est le temps approximatif d'exécution prévisible pour une liste de 80 000 éléments ?

2 points
La taille de la liste a été multiplié par 16, le temps sera donc multiplié approximativement par $16^2 = 256$, l'exécution devrait donc prendre environ $0,75 \times 256 = 192s$