Exercice 2 (4 points)

Cet exercice porte sur la programmation et les algorithmes de tri.

Au service des urgences d'un hôpital, le triage consiste à classifier ou à déterminer le degré de priorité des patients. Il implique une réévaluation périodique et systématique de ce degré pour les patients en attente.

Dans le système informatique, chaque patient obtient un identifiant à son arrivée en salle d'attente ainsi qu'une priorité dépendant de la gravité potentielle de ses symptômes. Le patient ayant la priorité 1 est le premier qui doit être pris en charge et deux patients ne peuvent pas avoir la même priorité.

Pour modéliser la salle d'attente en Python, chaque patient est représenté par un tuple composé de son identifiant d'arrivée et de sa priorité. Ainsi, la variable `attente` implémentée ci-dessous représente une salle d'attente de trois patients où, à cet instant, le patient identifié 47 sera le premier à être pris en charge, puis le patient 45 et finalement le patient 49.

```
attente = [(45,2),(47,1),(49,3)]
```

1. Écrire l'instruction qui permet d'insérer dans la liste `attente`, définie ci-dessus, un nouveau patient identifié 50 avec une priorité de 4.

2. Pour optimiser le traitement informatisé de prise en charge des patients, on veut que la salle d'attente soit ordonnée par priorité. On utilise alors la fonction `tri(attente)` donnée ci-dessous qui renvoie la salle d'attente triée dans l'ordre croissant des priorités.

```
def tri(attente) :
    for i in range(len(attente)) :
        pos = i
        mini = attente[i][1]
        for j in range(i, len(attente)) :
            if attente[j][1] < mini :
                pos = j
                mini = attente[j][1]
        temp = attente[i]
        attente[i]= attente[pos]
        attente[pos] = temp
```

   a. Quel algorithme de tri est ici implémenté ?
      - ✓ le tri fusion
      - ✓ le tri par insertion
      - ✓ le tri par sélection
      - ✓ le tri rapide

   b. Quelle est la complexité en temps des tris par insertion et par sélection ?
      - ✓ constante : $O(1)$
      - ✓ logarithmique : $O(\log n)$
      - ✓ linéaire : $O(n)$
      - ✓ quadratique : $O(n^2)$

3. On considère dans cette question que la salle d'attente est triée par ordre croissant des priorités.

   a. Quand un patient est pris en charge, il faut l'enlever de la salle d'attente. La fonction `quitte(attente)` supprime le patient de priorité 1 de la liste `attente` passée en paramètre. Cette fonction renvoie la nouvelle salle d'attente. On aurait par exemple :

```
>>> quitte([(47, 1), (45, 2), (49, 3)])
[(45, 2), (49, 3)]
```

   Recopier et compléter la ligne 14 en définissant une liste en compréhension.

```
13    def quitte(attente) :
14        return [. . . . . .]
```

   b. La fonction `quitte(attente)` ne met pas à jour les priorités des patients. Écrire une fonction `maj(attente)` qui met à jour, dans une nouvelle liste, les priorités des patients suite à la prise en charge d'un patient et renvoie cette nouvelle salle d'attente. On aurait par exemple :

```
>>> maj([45, 2), (49, 3)]
[(45, 1), (49, 2)]
```

4.

   a. Écrire la fonction `priorite(attente, p)` qui renvoie l'ordre de priorité d'un patient `p` de la liste `attente`. On aurait par exemple :

```
>>> priorite([(45,2),(47,1),(49,3)], 49)
3
```

   b. La fonction `revise(attente,p)` renvoie une nouvelle salle d'attente non triée où le patient `p` dont la priorité médicale a évolué reçoit la priorité 1 et les priorités des autres patients sont mises à jour. On aurait par exemple :

```
>>> revise([(45,2),(47,1),(49,3)], 49)
[(45,3),(47,2),(49,1)]
```

   Recopier et compléter les lignes 28, 29 du code ci-dessous.

```
23 def revise(attente, p) :
24     nouvelle = []
25     n = priorite(attente, p)
26     for (patient, prio) in attente :
27         if patient == p :
28             nouvelle.append( ..... )
29         elif ..... :
30             nouvelle.append((patient, prio + 1))
31         else :
32             nouvelle.append((patient, prio))
33     return nouvelle
```