Métropole – 2025 – sujet1 - Correction

Exercice 2 (6 points)

---

1. Instanciation des taches

```
tache1 = Tache ( 1, "Répondre aux e-mails", 45 )

tache2 = Tache ( 2, "Ranger ma chambre", 60 )
```

Commentaire : la durée restante est initialisée automatiquement à la durée totale lors de la création de l'objet.

2. Méthode avancer

```
def avancer ( self, n ) : self.duree_restante = self.duree_restante - n
```

Commentaire : avancer de n minutes revient à diminuer la durée restante.

3. Méthode est_terminee

```
def est_terminee ( self ) : return self.duree_restante <= 0
```

Commentaire : une tâche est terminée dès que la durée restante est nulle ou négative.

4. Ajouts dans la file de priorité

Après ajout de la tâche 6 (priorité 2) puis de la tâche 7 (priorité 4), la file devient :
`[début] (<t3>,4) (<t7>,4) (<t1>,3) (<t2>,3) (<t6>,2) (<t4>,1) (<t5>,1) [fin]`

5. defiler et examiner

`f.defiler()[0]` vaut <t3>.
Après defiler :
`[début] (<t1>,3) (<t2>,3) (<t4>,1) (<t5>,1) [fin]`

`f.examiner()[1]` vaut 4.
La file reste inchangée.

6. Fonction ajouter_file_prio

```
def ajouter_file_prio(f, t, p):
    f_aux = File()

    while not f.est_vide() and f.examiner()[1] >= p:
        f_aux.enfiler(f.defiler())

    f_aux.enfiler((t, p))

    while not f.est_vide():
        f_aux.enfiler(f.defiler())

    while not f_aux.est_vide():
        f.enfiler(f_aux.defiler())
```

Commentaire : on conserve l'ordre relatif des tâches de même priorité.

- La **première boucle** déplace les éléments de priorité **supérieure ou égale** à `p` dans la file auxiliaire.
- On insère ensuite `(t, p)` **au bon endroit**.
- La **seconde boucle** vide le reste de `f` dans `f_aux`.
- La **dernière boucle** remet tous les éléments dans `f` en conservant l'ordre.

7. Coût temporel

Dans le pire des cas, chaque élément de la file est déplacé une fois.
Le coût temporel est donc O(n), où n est le nombre d'éléments de la file.

8. Fonction planning (Pomodoro)

```
def planning(f):
    ordre = []

    while not f.est_vide():
        t, p = f.defiler()
        t.avancer(25)
        ordre.append(t)

        if not t.est_terminee():
            ajouter_file_prio(f, t, p)

    return ordre
```

Commentaire : chaque tâche apparaît autant de fois que nécessaire jusqu'à sa complétion par tranches de 25 minutes.