Test 1 Informatique : Corrigé

```
import random

# Question 1 - Classe Task
class Task:
    def __init__(self, name, duration):
        self.name = name
        self.duration = duration

    def __str__(self):
        return "Task '{}' - Duration: {}".format(self.name, self.duration)

    def __del__(self):
        print("Fin d'exécution de la tâche '{}'".format(self.name))

# Question 2 - Classe TaskQueue
class TaskQueue:
    def __init__(self):
        self.queue = []

    def enqueue(self, task):
        self.queue.append(task)

    def dequeue(self):
        if self.queue:
            return self.queue.pop(0)
        return None

    def __str__(self):
        return "File de tâches: {}".format([str(task) for task in self.queue])

# Question 3 - Programme principal
if __name__ == "__main__":
    # 3a
    DURATION = random.uniform(1e-9, 1e-6)
    # 3b
    task_queue = TaskQueue()
    # 3c
    n = random.randint(3, 10)
    for i in range(n):
        task = Task("Task-{}".format(i), random.uniform(1e-6, 1e-3))
        task_queue.enqueue(task)

    print("État initial de la file:", task_queue)

    # 3d - Simulation du traitement de tâches
    def execute(task):
        if hasattr(task, 'finished'):
            return
        else:
            task.duration -= DURATION
            if task.duration < DURATION :
                task.finished = True
            execute(task)

    # 3e
```

```python
total_time = 0
for _ in range(n):
    task = task_queue.dequeue()
    total_time += task.duration
    task_queue.enqueue(task)
print("Temps total nécessaire pour traiter toutes les tâches dans la file:", total_time)

# 3f
for _ in range(n):
    current_task = task_queue.dequeue()
    execute(current_task)
    if hasattr(current_task, 'finished'):
        del current_task

# 4a - (modification de la classe Task pour inclure la priorité)
class Task:
    def __init__(self, name, duration, priority=0):
        self.name = name
        self.duration = duration
        self.priority = priority
    def __str__(self):
        return "Task '{}' - Duration: {} - Priority: {}".format(self.name, self.duration, self.priority)

    def __del__(self):
        print("Fin d'exécution de la tâche '{}'".format(self.name))

# 4b (implémentation de la classe PriorityTaskQueue)
class PriorityTaskQueue(TaskQueue):
    def enqueue(self, task):
        # Trie la file en fonction de la priorité (ordre décroissant)
        self.queue.append(task)
        self.queue.sort(key=lambda x: x.priority, reverse=True)

# 4c : update 3b and 3c
task_queue = PriorityTaskQueue()

# Génération aléatoire de n tâches avec priorité
for i in range(n):
    task = Task("Task-{}".format(i), random.uniform(1e-6, 1e-3), random.randint(1, 5))
    task_queue.enqueue(task)

print("État initial de la file de priorité:", task_queue)
```