1. Une file d'attente prioritaire est une file d'attente dans laquelle chaque élément a une priorité qui affecte l'ordre de traitement des éléments dans une fille comme suit :
   * Un élément de priorité supérieure est traité avant tout élément de priorité inférieure, même s’il a été ajouté après dans la file.
   * Deux éléments de même priorité sont traités selon l'ordre dans lequel ils ont été ajoutés (FIFO).
2. Définir une structure de données pour représenter une file d'attente prioritaire. Supposons que la valeur de priorité est comprise entre 0 et n-1, où n-1 est la priorité la plus élevée et 0 la plus basse. Ici, on vous demande de proposer deux implémentations différentes. Une implémentation en utilisant une seule file et une deuxième en utilisant plusieurs files.
3. Pour chacune des deux implémentations, écrire la fonction **Enfiler** qui prend en entrée un élément et sa priorité et l'ajoute à la file d'attente prioritaire. Quelle est la complexité de votre fonction dans les deux cas ?
4. Pour chacune des deux implémentations, écrire la fonction **Défiler** qui supprime et retourne un élément de la file d'attente. Quelle est la complexité de votre fonction dans les deux cas ?
5. Ecrire un programme principal qui teste l’implémentation ci-dessus.
6. A priority queue is a queue in which each element has a priority that affects the order of processing the elements in the queue as follows:
   * An element with higher priority is processed before any element with lower priority, even if it was added later to the queue.
   * Two elements with the same priority are processed in the order they were added (FIFO).
7. Define a data structure to represent a priority queue. Assume that the priority values range from 0 to n-1, where n-1 represents the highest priority and 0 represents the lowest priority. Here, you are asked to propose two different implementations. One implementation using a single queue and another implementation using multiple queues.
8. For each of the two implementations, write the "Enqueue" function that takes an element and its priority as input and adds it to the priority queue. What is the complexity of your function in both cases?
9. For each of the two implementations, write the "Dequeue" function that removes and returns an element from the priority queue. What is the complexity of your function in both cases?
10. Write a main program that tests the above implementation.