Please create a generic priority queue in your favorite language.  Specifications for required and optional functionality can be found below.  If any requirements are unclear, please feel free to reach out and ask for clarification.

Code a generic Priority Queue

* Priority is UInts
* Number of priorities bound by the limits of UInts
* Higher priorities are returned first
* Items within a given priority will be returned in FIFO order.
* Support the Enqueue, Dequeue and Count methods

Problem focus

* Solve for performance
* Write unit tests that validate the items are returned in the order expected.

Extra Credit:

* Make thread safe
* Write a Count(unit priority) method.

**Solution:**

The priority queue is implemented with a Binary Heap data structure. The binary heap structure is implemented using vector to hold jobs with their priorities. For each element at position i in the vector, the left child node of the heap will be at position 2\*i+1 and right child node will be at position 2\*i+2. The parent node will also hold a job of higher priority or equal priority than its child nodes. The first element in the vector will contain the job with highest priority in the heap. New jobs are added to end of the vector and “percolated up” the tree such that heap order is maintained. As highest priority job is removed from top the heap, the nodes are shifted down till the heap order is restored.

The tree maintains heap order such that jobs of similar priority level are arranged by order of insertion such that are retrieved in FIFO order.

**Performance:** Insertion and Deletion of elements (along with operations to maintain heap order) have a time complexity of O(Log N) where N is the number of jobs in the job. The insertion and deletion operations have time complexity of O(1) for vectors and restoration of heap order has time complexity of O(Log N) in worst case.

**Thread Safety:** Critical section are used to ensure thread safety during insertion and deletion operations.

**Count(priority) :** Implemented in code. Searches through the heap till it find jobs of higher or equal priority.

Future Improvement: The current implementation, contains a node each job which means that the heap can contain jobs of duplicate priority. A variation on this implementation is not allow duplicate priorities in tree and store job of similar priority in a FIFO structure. For this implementation, that would mean modifying the Job class to have queue of jobs.