### Programming Guide for “Heaps and Priority Queues”

#### 1. MinHeap Class Implementation

**Overview:** A MinHeap is a complete binary tree where each node is smaller than its children. The root, therefore, is the minimum element in the tree.

**Key Operations:** - insert: Adds a new element to the heap. - extract\_min: Removes and returns the minimum element from the heap. - get\_min: Returns the minimum element without removing it.

**Pseudo Code for insert:**

FUNCTION insert(element)  
 ADD element to the end of the heap  
 BUBBLE UP the element to maintain the heap property  
END FUNCTION

**Pseudo Code for extract\_min:**

FUNCTION extract\_min()  
 IF heap is empty  
 RETURN None  
 SWAP the first element with the last element  
 REMOVE the last element (minimum element)  
 BUBBLE DOWN the new first element to maintain heap property  
 RETURN the removed element  
END FUNCTION

**Pseudo Code for get\_min:**

FUNCTION get\_min()  
 RETURN the first element of the heap (minimum) IF heap is not empty  
 ELSE RETURN None  
END FUNCTION

**Implementation Tips:** - Use a list to store elements of the heap. - For a node at index i, its children are at indices 2\*i + 1 and 2\*i + 2, and its parent is at (i-1) // 2. - “Bubbling up” and “bubbling down” are key to maintaining the heap property after insertion and deletion.

#### 2. Finding the kth Largest Element

**Overview:** This task involves finding the kth largest element in an unsorted array.

**Pseudo Code:**

FUNCTION find\_kth\_largest(nums, k)  
 USE a heap to store the k largest elements  
 FOR each element in nums  
 ADD element to the heap  
 IF size of heap exceeds k  
 REMOVE the minimum element from the heap  
 RETURN the minimum element from the heap (kth largest)  
END FUNCTION

**Implementation Tips:** - Utilize Python’s heapq module for efficient heap operations. - Maintain a min-heap of size k. Once the size exceeds k, remove the smallest element. This ensures that you’re always keeping the k largest elements. - The root of the heap will be the kth largest element.