Question (Heap-sort)

We are given an unsorted array *A[1…n]*. Now, imagine its sorted version. The unsorted array has the property that each element has a distance of at most *k* positions, where $0<k<=n$, from its index in the sorted version. For example, when *k* is *2*, an element at index *5* in the sorted array, can be at one of the indices *{3,4,5,6,7}* in the unsorted array. The unsorted array can be sorted efficiently by utilizing a Min-Heap data structure. The outline of the algorithm isgiven below

- Create a Min Heap of size *k+1* with first *k+1* elements,
- One by one remove min element from the heap, put it in the result array, and add a new element to the heap from remaining elements.

a. Write down the complete algorithm in pseudocode convention to sort the array *A*.

b. Provide a tight asymptotic upper bound time complexity for this algorithm. Show your work.

c. Implement your solution in any language you prefer. Generate the array sizes of 100,1000,10000,100000 with using your student id as a seed. Run your solution and note the running times. Show your results.

You are expected to deliver a report and code about your solution.