Problem 1: Some algorithms textbooks have statements of the type

Every comparison-based sorting algorithm takes at least \( O(n \log n) \) time.

Does such a statement make sense?

Problem 2: Let \( A[1 \ldots n] \) be an array storing \( n \) numbers. In the January 25 lecture, we have seen algorithm \textsc{BuildHeap}(A) that rearranges the numbers in the input array \( A \) such that the resulting array is a max-heap; see page 56 of my handwritten notes. This algorithm uses the \textsc{Heapify}-procedure as a subrouting; see page 53 of my handwritten notes. Consider the following variant of this algorithm:

```
Algorithm \textsc{BuildHeap}'(A):
  for \( i = 1 \) to \( \lfloor n/2 \rfloor \) 
  do \textsc{Heapify}(A, i)
endfor
```

Give an example of an array \( A[1 \ldots n] \), where \( n \) is a small integer (such as \( n = 7 \)), which shows that algorithm \textsc{BuildHeap}' may not result in a max-heap.

Problem 3: Let \( A[1 \ldots n] \) be an array storing \( n \) pairwise distinct numbers, and let \( k \) be an integer with \( 0 \leq k < n \). We say that this array is \( k \)-sorted, if for each \( i \) with \( 1 \leq i \leq n \), the entry \( A[i] \) is at most \( k \) positions away from its position in the sorted order.

For example, a sorted array is 0-sorted. As another example, the array

\[
A[1 \ldots 10] = [1, 4, 5, 2, 3, 7, 8, 6, 10, 9]
\]

is 2-sorted, because each entry \( A[i] \) is at most 2 positions away from its position in the sorted order. For \( i = 3 \), \( A[3] \) is 2 positions away from its position, 5, in the sorted array. For \( i = 9 \), \( A[9] \) is 1 position away from its position, 10, in the sorted array.

Describe an algorithm \textsc{Sort} that has the following specification:

```
Algorithm \textsc{Sort}(A, k):
Input: An array \( A[1 \ldots n] \) of \( n \) pairwise distinct numbers and an integer \( k \) with \( 2 \leq k < n \). This array is \( k \)-sorted.
Output: An array \( B[1 \ldots n] \) containing the same numbers as the input array. The array \( B \) is sorted.
Running time: Must be \( O(n \log k) \).
```

Explain why your algorithm is correct and why the running time is \( O(n \log k) \).

Hint: Use a min-heap of a certain size.