Suppose you work on a droid assembly line. You have a supposedly sorted array of \( N \) Droid objects that implement Comparable. However, when looking through your array, you realize these aren’t the droids you’re looking for! The machine malfunctioned and made at most \( k \) mistakes: there are no more than \( k \) inversions, where we define an inversion as a pair of droids that is not in the right order.

*Hint:* The array \([0 1 1 2 3 4 8 6 9 5 7]\) has 6 inversions: 8–6, 8–5, 8–7, 6–5, 9–5, 9–7.

For the questions below, give the typical or expected runtime. For example, for quicksort, assume that the pivot choices result in \( O(\log N) \) recursive depth.

1. For each \( k \), give the most efficient sorting algorithm and its simplified asymptotic runtime.

   (a) \( k \in O(N) \)
   
   Algorithm: Insertion sort
   
   Runtime: \( \Theta(N) \)

   (b) \( k \in O(N^2) \)
   
   Algorithm: Merge sort, quicksort, or heapsort
   
   Runtime: \( \Theta(N \log N) \)

   (c) \( k \in O(\log N) \)
   
   Algorithm: Insertion sort
   
   Runtime: \( \Theta(N) \)

2. Two weeks later, you’re given another batch of droids that are supposed to be sorted on a 32-bit int ID, an instance variable of Droid. The machine hasn’t been fixed and again made at most \( k \) mistakes. For each \( k \), give the most efficient sorting algorithm and its simplified asymptotic runtime.

   (a) \( k \in O(N^2) \)
   
   Algorithm: Radix sort
   
   Runtime: \( \Theta(N) \)

   (b) \( k \leq 5 \)
   
   Algorithm: Insertion sort or radix sort
   
   Runtime: \( \Theta(N) \)