(1). Finding the k^{th} smallest element is a generalized case of the linear median finding problem, which is called "QuickSelect."

kthsmallest.cpp differs from the QuickSort algorithm because whereas QuickSort will recursively work on two partitions, kthsmallest.cpp will only recursively work on the partition where the k^{th} element lies.

kthsmallest.cpp has the same issue with potential n^2 runtime complexity blow-up, if the input array is already sorted, and we are searching for a maximal element. However, the algorithm should work in linear time on average, according to this analysis found online.

(2). There are technically two base cases of kthsmallest.

1. $k =$ pi. This is the case where the pivot chosen is equal to k.
2. low $\geq$ high. This is the case where the partition's left bound is greater than or equal to its right bound. In this case, we return the left bound as the k^{th} element.