Problem One:

For this algorithm, I thought that the kth smallest value would mean that it is k spaces from the start of the list. However, I also had to take into account that the index for lists starts at 0. So, if I were to find the smallest value I would enter k as one but if k was my index I would get the second number back because its index is one. From here I changed the return of the function from numList[k] to numList[k-1] because it assures that I will get the right value for the kth smallest. I also added in an if statement to make sure the user entered a valid number for k because otherwise, it would not function correctly.

T(n) = 1. The function is constant. Index has O(1) complexity and at most is operated once because of the if-else statement

Upper Bound = 2. It's just constantly bigger than T(n)

Big O Notation = O(1)

Problem Two:

For this algorithm, I started knowing I would use a nested for loop to iterate through all the numbers. From here, I decided to create a variable 'found' and set it to false. This will be used to determine if we found two numbers that equaled 25. Next, I populated the inside of the inner for loop with an if statement that checks if the two numbers sum to 25. If they do, then found will be set to true. After iterating through the possible combinations, the function will return the boolean value of found. If it is false, that means there were not two numbers in the list that summed to 25. It it returned true then that means there were two numbers that summed to 25.

T(n) = n^2 +1. There's one assignment in the inner for loop and since it's a nested for loop you get 1*n*n. This gives you n^2 and then you add the assignment from outside the loop to get n^2+1

Upper Bound = 2n^2

Big O Notation = O(n^2)