Looking at the problem at first it appeared to be very simple. And in the end it really wasn't that bad, but the complexity needing to be O(log n + k) made it a little harder. My first thought was to simply go through the list and compare each value, this would allow the range to be selected even in an out of order list(which for a while I thought was a part of the problem). This however was a complexity of O(n) because it would have to check every item in the list, so I figured that I would need to implement a search algorithm that works in O(log n) complexity to find the high and low points. And after a look in my notes I found that I should implement a binary search.

When implementing the algorithm I first started by filling in all of the base cases. The ones I identified were to return none if list was none or if lo > hi. And then to return an empty string even if the list is empty and then the problem came if High or low were None. I combatted this by setting low to negative infinity if it was none and setting high to infinity if it was none, this way all of the data in the list would have to be above or below high or low. The next thing to implement was the search algorithm. I had to implement two different basic binary search algorithms. They are only ever so slightly different. One has a <= when comparing the value of the list at the middle and the target. I came to this while bug fixing. I'm not entirely sure why it fixed it to be honest, but I was having an issue with the edge cases. The binary search algorithms both work as expected, cutting the range of elements in half until the value or closest value is found. When the high and low values are found those variables are returned, limiting the range of the list in the line "return list[left:right]".

Overall this lab went well for me and didn't take much time once I knew what direction I was going.