# Exercise 3

1. Interpolation search looks for where the item is likely to occur, which could be before, after, or at the midpoint, while Binary search always starts at the midpoint and discards a half chunk of data until the element it is searching for is found. This makes Interpolation search more efficient because it starts closer to the target item.  
   Interpolation search has an average time complexity of O(log(log(n))) which is faster and more efficient than the average time complexity of Binary search, O(log(n)).
2. If the data follows a different distribution than a uniform distribution, then the performance will be affected. This is because anything other than a uniform distribution will cause the estimation of the target element to be inaccurate. If the estimation of the target element is inaccurate, then the search space becomes larger and in return the search time becomes longer. This causes inefficiencies, so Interpolation search is best used when the array is uniformly distributed.
3. The part of the code in the question which would be affected is the variable pos. This is because the pos variable is set to look for the position of the target element in a uniformly distributed array. Changing the distribution, the value of the pos variable would have to be changed.
4. A) Linear search is the only option when the data is not sorted. This is because Binary and Interpolation search require the data to be sorted.  
   B) Linear search will outperform Binary search and Interpolation search when the data has duplicates. It outperforms when the data has duplicates because Binary search assumes there are no duplicates, because duplicates would cause the search space to not be cut in half each time a binary search is conducted, making it slower. For Interpolation search, if there are many duplicates the data becomes less uniform, which will cause the search position to be less accurate, making the search algorithm slower. One way to improve Binary or Interpolation searches would be to start iterating through the array to find the target element while storing each value in a dictionary cache, then when a value that is a duplicate is found, it gets skipped since the function recognizes the value already in the cache and moves to the next value.