Assignment 03

(20 points each)
Submit a text file, Java, Python, C or C++ source file or .ipynb notebook file.

1. Describe the worst case data and the best case data for each of the following sorting algorithms. Also, include the big O notation for each case.
    - Bubble Sort
    - Selection Sort
    - Insertion Sort
    - Merge Sort
    - Quicksort

2. Implement the insertion sort function.

3. Write a function that accepts an array of size n containing random numbers ranging from 0 to n-1.
    - This array may contain duplicates
    - The numbers are not arranged in any particular order

This function should return a new array that consists of any missing numbers within the range from 0 to n-1.
This function must have a time complexity of O(n) to get full credit.

For example:
Given the array `[0, 3, 6, 7, 3, 3, 0, 4]`, this function should return `[ 1, 2, 5 ]`

4. Write a function that returns the first non-repeating character in a string with O(n) efficiency. It should return none or null if there are no non-repeating conesutive characters.

For example:

- string `"aaaaabbbbbbc"`, this function should return `"c"`
- string `"aabab"` should return `"b"`
- string "`aababb`" should return None ("b" is repeating)

5. Write a function that given an array of integers and a target value, returns the length of the longest subarray with a sum equal to the target value. Write the function with O(n) efficiency for full credit.
Note: while the sliding window technique is acceptable as a solution, try solving this using a hash table.

For example:
Given an array `[3, 1, -1, 2, -1, 5, -2, 3]` and a target value of `3`, the longest subarray length is `5` (`[-1, 2, -1, 5, -2]`)