BIG O NOTATION PRACTICE

Problem 1

We can use the following algorithm to see if a numerical value exists in an array. The algorithm will find the first place where the number exists and return its index.

```
findFirstIndexOfNumber(number, array) {
  for (i = 0; i < array.length; i++) {
    if (array[i] == number) {
      return i;
    }
  }
  return -1;
}
```

What is the Big O runtime of this algorithm?

Problem 2

Now, instead of finding just the first index where a number exists, our algorithm needs to find *every* index where the number exists. Here's the algorithm:

```
findEachIndexOfNumber(number,array) {
  arrayOfIndexes = [];
  for (i = 0; i < array.length; i++) {
      if (array[i] == number) {
          arrayOfIndexes.add(i);
    }
  }
  return arrayOfIndexes;
}
```

What is the Big O runtime for this algorithm?

Problem 3

The following function checks to see if the last item in a data set is higher or lower than the first item in a data set — and returns *Higher*, *Lower*, or *Neither*.

```
array = [36, 14, 1, 7, 21];

higherOrLower(array) {
  if (array[array.length -1 ] > array[0]) {
    return "Higher";
  else if (array[array.length -1 ] < array[0]) {
    return "Lower";
  } else {
    return "Neither";
  }
}
```

What is its Big O?

Problem 4

We can use the following function to determine the sum of an array of sequential numbers:

```
array = [1,2,3,4,5,6,7,8];

determineSumOfSequentialArray(array) {
   sum = 0;
   for (let i = 0; i < array.length; i++) {
       sum += array[i];
   }
   return sum;
}
```

What is the Big O notation for this algorithm?