1) You have two arrays. One of prime numbers and one of non-primes. Write a function that adds the numbers in the arrays together, but add the non-primes in reverse order, and returns that information in a new array. So the first prime should be added to the last non-prime, the second prime should be added to the second to last non-prime, etc.

```
var primes = [2, 3, 5, 7, 11, 13, 17, 19];
var nonPrimes = [1, 4, 6, 8, 9, 10, 12, 14];
```

2) Using the following arrays, write a function that merges the numbers together, in order, and returns the result as a new array. The result should be [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].

   Hint: ***Array.prototype.Sort*** can do the sorting for you, based on a comparison function.

   Write a comparison function, that takes two parameters. If the first is less than the second, return -1. If the first is greater than the second, return 1. If they are equal, return 0.

   Pass your comparison function to Array.prototype.Sort.

```
var numbers1 = [4, 1, 6, 5, 8];
var numbers2 = [7, 3, 2, 9, 10];
```

3) Create a constructor for an object called ***rightTriangle***. The constructor should take in two parameters, ***a***, ***b.***

   Give the triangle the properties ***a***, ***b***, ***c***.

   Now, create a prototyped function called ***hypotenuse***.

   When called, it should square a, square b, add them together, and set c equal to the square root of their sum.

   Create a new ***rightTriangle*** object, pass in 3 and 4 for the parameters. When you call the object's ***hypotenuse*** function, ***c*** should be equal to 5. Call console.log on your object's ***c*** and see if it is 5.

4) Write a function that takes a number as a parameter. If the number is 0, just return 0, otherwise return the number.

   Now we're going to use something called ***recursion***. A function that calls itself to do some work on a new value is recursive.

Instead of returning the number, return the number plus the function called on the number - 1. For example, `functionName(number-1)`.

When called, your new function adds up numbers from 1 to any given number, plus every number below it. For example, an input of 5 will return 1 + 2 + 3 + 4 + 5 = 15.