- Define a function named `allIndexesOf` that takes in two arguments. The first argument should be the array to search and the second argument should be the value you want to search for. If the item does not exist in the provided array, return an empty array.

  Given:

  var `fruits = [`"apple", "banana", "orange", "apple", "pineapple"`];`

  - `allIndexesOf(fruits, "apple")` should return the array [0, 3]
  - `allIndexesOf(fruits, "guava")` should return the array []
  - `allIndexesOf(fruits, "pineapple")` should return [4]

- Define a function named `removeAll(array, value)` that takes in two arguments. The first argument should be an array and the second argument should be a value you wish to remove

  Given:

  var `bugs = [`"mosquito", "ant", "scorpion", "ant", "ant", "mosquito", "typo", "reference error", "type error"`];`

  - `removeAll(bugs, "ant")` should return `["mosquito", "scorpion", "mosquito", "typo", "reference error", "type error"]`
  - `removeAll(bugs, "mosquito")` should return `["ant", "scorpion", "ant",   "ant", "typo", "reference error", "type error"]`
  - `removeAll(bugs, "roach")` should return the original array b/c "roach" has no occurrances.

- Make a function called randomIntBetween(min, max) that returns a random number between the min and the max.
- Make a function called `coinFlip()` that returns either 0 or 1, randomly
- Make a function called `twoDice()` that returns the sum of rolling two six sided dice.
- Make a function called `twentySidedDie()` that returns a random integer between 1 and 20.
- Make a function called `twelveSidedDie()` that returns a random integer between 1 and 12.
- Make a function called `tetrahedron()` that returns a random integer between 1 and 4.

- Make a function called `rollDie()` that returns an integer between 1 and 6.
- Make a function called `listOfRolls(num)` that takes in a number containing how many 6-sided dice rolls you want to make. The `listOfRolls` function should return an array of that length, where each element of the array is the result of the `rollDie` function.
- Make a function called `listOfRollsFromDieFunc(numberOfRolls, diceFunction)`

  This function should take in two arguments:

  - The first argument is the number of rolls you want to make.
  - The second argument is a function that contains the function definition for the type of die you want to roll.

  For example, if we call `listOfDieRollsFromDieFunc(1, tetrahedron)` , then the function will return an array containing one value that is the result of calling the `tetrahedron` function.

```
/**
 * JS Array Practice
 * Intermediate Array practice: array creation, iteration, and manipulation
 */

// Exercise 0. Write a function named first() that returns only the first element of an array

// Exercise 1. Write a function named secondToLast() that returns the second to last element

// Exercise 2. Write a function named rest() that takes an an array and returns an array containing
everything except the first element.

// Exercise 3. Write a function named getLongestString that takes in an array of strings and returns
the longest string of that array

// Exercise 3.1 Write a function named getShortestString that takes in an array of strings and returns
the shortest string in that array.

// Exercise 4. Write a function named addTwoArrays that takes in two, one dimensional arrays. The
function should return a single array containing all of the elements of the first array along with all of
the elements of the second array
// Example: addTwoArrays([1, 2, 3], [4, 5, 6]) should return [1, 2, 3, 4, 5, 6]

// Exercise 5. Write a function named getUniqueValues that takes in an array and returns the array
without any duplicates
// Example: getUniqueValues(["a", "b", "a", "b", "c", "c"]) should return ["a", "b", "c"]

// Exercise 6. Write a function named reverseArray that takes in an array and returns it reversed, but
without altering the original array.

// Exercies 7. Write a function named getRandomQuote().
//   Inside of the function, create an array of strings where each string is a quote or thought you find
inspirational
//   getRandomQuote should generate a random number between 0 and the array's length minus 1
//   use the randomly generated number as your index
//   return a random quote.

// Exercise 8. Write a function named getIndexesOf() that takes in two arguments.
// The first argument should be a specific numeral or character
// The second argument should be any given string
// getIndexesOf() should return an array containing all of the indexes of that character in the string
// Example: getIndexesOf("a", "banana") should return the array [1, 3, 5]
// Example: getIndexesOf("z", "banana") should return an empty array [] since there are no "z"
characters in "banana"

// Exercise 9. Write a function named removeAll.
// It should accept an array and a value
// removeAll should return an array with all of the original contents EXCEPT for the provided value
// iterate across the input array
// output array
// Example: removeAll([1, 2, 3], 2) should return [1, 3]
// Example 2: removeAll([2, 2, 3, 4, 5, 2, 2], 2) should return [3, 4, 5]
```

// Exercise 10. Write a function named firstTenFibonacciNumbers() that returns an array of the first ten fibonacci numbers

// Exercise 11. Write a function named getNFibonacci(n) that returns an array containing the first n fibonacci numbers

// Exercise 12. Write a function named moveFirstToLast() that takes in an array
// the function should return the array with the first element at the end
// Example: moveFirstToLast([1, 2, 3, 4]) should return [2, 3, 4, 1]

// Exercise 13. Write a function named zip() that takes in two arrays with the same number of elements
// Zip returns a new array of arrays where each element is an array of the two elements at the same index
// Example: zip([1, 2, 3], [4, 5, 6]) returns [[1, 4], [2, 5], [3, 6])
// Example: zip(["a", "b", "c"], ["x", "y", "z"]) returns [["a", "x"], ["b", "y"], ["c", "z"]]