Using *methods*, write functions to…

1. Take in a number and return the square root of that number.

```
function sqRt(num) {
    return Math. sqrt(num);
}
console.log(sqRt(144)); // output 12
```

2. Take in a string, replace the first instance of the string ‘e’ within the string with the character ‘*’, return the modified string
   E.g. “Eeyore” -> “E*yore”
   “Banana” -> “Banana”
   “Apple” -> “Appl*”

```
function replaceE(string) {
    let firstLetter = string.slice(0,1);
    let newStr = string.substring(1);
    let partTwo = newStr.replace(/e{1,}/,"*");
    console.log(firstLetter + partTwo);

}
replaceE("Eeyore"); // E*yore
replaceE("Banana"); // Banana
replaceE("Apple"); //  Appl*
```

3. Take in a number, return the number of digits in that number **without doing any dividing**
   a. Hint: your solution need not work for really big numbers
   b. Hint 2: check out the concept called “casting”

```
function nmbrOfDigs(num) {
function nmbrOfDigs(num) {
   return Math.log(num) * Math.LOG10E + 1 | 0;
}
console.log(nmbrOfDigs(10)); // 2
console.log(nmbrOfDigs(224234)); // 6
```

4. Take in three numbers: day, month and year. Create a Date object with these values. Then, determine if that Date is part of the weekend. If it is, return true. Otherwise, return false.
   a. Note: do **NOT** build your own Date object. Use the built-in Date object.
      e.g. weekendChecker(1,31,2020) -> false
      January 31, 2020 is **not** part of the weekend

```
function isWeekend(year,month,day) {
    let date = new Date(year,month,day)
    console.log("The date you entered is: " + date + ". Was it the weekend?");
    if (date.getDay() === 6 || date.getDay() === 0) {
        return true;
    } else {
        return false;
    }
};
console.log(isWeekend(1985,9,20));
// The date you entered is: Sun Oct 20 1985 00:00:00 GMT-0600 (Mountain Daylight
Time). Was it the weekend?
// true
```

Given the following code…

```
var programming = {
      languages: ["JavaScript", "Python", "Ruby"],
      isChallenging: true,
      isRewarding: true,
      difficulty: 8,
      jokes:
"http://stackoverflow.com/questions/234075/what-is-your-best-programmer-jok
e"
};
```

5. Write the command to add the language "Go" to the end of the languages array.

```
programming.languages.push("Go");
console.log(programming.languages); // ["JavaScript", "Python", "Ruby", "Go"]
```

6. Change the difficulty to the value of 7.

```
programming.difficulty=7;
console.log(programming.difficulty); // 7
```

7. Using the delete keyword, write the command to remove the jokes key from the programming object.

```
delete programming.jokes;
console.log(programming.jokes); // undefined
```

8. Write the command to add a new key called isFun and a value of true to the programming object.

```
programming.isFun = true;
console.log(programming.isFun); // true
console.log(programming); // {languages: Array(4), isChallenging: true,
isRewarding: true, difficulty: 7, isFun: true}
```

9. Using a loop, iterate through the languages array and console.log all of the languages.

```
for (i=0; i<programming.languages.length; i++) {
    console.log(programming.languages[i]);
}
// JavaScript
// Python
// Ruby
// Go
```

10. Using a loop, console.log all of the keys in the programming object.

```
for (let [key, value] of Object.entries(programming)) {
    console.log(key);
};
// languages
// isChallenging
// isRewarding
// difficulty
// isFun

// THIS COMMAND ALSO WORKS, BUT DOESN'T USE A LOOP >>>>>
console.log(Object.keys(programming));
// (5) ["languages", "isChallenging", "isRewarding", "difficulty", "isFun"]
```

11. Using a loop, console.log all of the values in the programming object.

```
for (let [key, value] of Object.entries(programming)) {
    console.log(value);
};
// (4) ["JavaScript", "Python", "Ruby", "Go"]
```

```
// true
// true
// 7
// true
```

12. Write some essential components of a to-do list maker in three parts:
    a. Write a function that takes in two inputs: a string containing a task and a string representing a date containing a deadline for completion. The function returns a custom object called 'Item' containing the fields 'task' and 'deadline'.
    E.g.
    Item('get groceries', ''March 10, 2020") -> {task: 'get groceries', deadline: 'March 10, 2020')

    You may use a Date object to represent the deadline if you prefer.

```
function taskDeadline (task,deadline) {
    this.task = task;
    this.deadline = deadline;
}
let item = taskDeadline("get groceries", "July 16,2020")
console.log(taskDeadline);

// this isn't printing the object in the way the
instructions are asking for
// this is only printing the object constructor
// try this one again using a factory function maybe?
```

    b. Write a function that takes in an array of custom Item objects created in part A, iterates through the array, and returns a formatted string containing an html element of an unordered list where each list item contains the information stored in a single Item object from the input array. The output should be valid HTML.

    c. Test your functions by creating a list of three or so Item objects (using the function from part A), then taking the HTML produced from part B and adding it to the HTML in your document. *Consider using the methods **createElement** and **appendChild** in your approach.*