1. Bean counting

You can get the Nth character, or letter, from a string by writing "string"[N]. The returned value will be a string containing only one character (for example, "b"). The first character has position 0, which causes the last one to be found at position string.length - 1. In other words, a two-character string has length 2, and its characters have positions 0 and 1.

Write a function countBs that takes a string as its only argument and returns a number that indicates how many uppercase “B” characters there are in the string.

Next, write a function called countChar that behaves like countBs, except it takes a second argument that indicates the character that is to be counted (rather than counting only uppercase “B” characters). Rewrite countBs to make use of this new function.

// Your code here.

console.log(countBs("BBC"));

// → 2

console.log(countChar("kakkerlak", "k"));

// → 4

Solution:

Source Code:

```
const countBs=(str)=> {
    let count = 0;
    for (let i = 0; i < str.length; i += 1) {
      if (str.charAt(i) === "B")
        count += 1;
    }
    return count;
  }

  const countChar=(str, element) =>{
    let count = 0;
    for (let i = 0; i < str.length; i += 1) {
      if (str.charAt(i) === element)
        count += 1;
    }
    return count;
  }
```

```
  console.log(countBs("BBC"));
  // → 2
  console.log(countChar("kakkerlak", "k"));
//    → 4
```

Output:

```
PROBLEMS    OUTPUT    TERMINAL    COMMENTS    DEBUG CONSOLE

PS G:\programs\node js> node beanCount.js
2
4
PS G:\programs\node js>
```

2. Minimum

Write a function min that takes two arguments and returns their minimum.

// Your code here.

console.log(min(0, 10));

// → 0

console.log(min(0, -10));

// → -10

Solution:

Source Code:

```
const min=(a,b)=>{
    return (a<=b)? a: b;
}

console.log(min(0,10));
console.log(min(0,-10));
```

Output:

```
PS G:\programs\node js> node min_num.js
0
-10
PS G:\programs\node js>
```

3. Looping a triangle

Write a loop that makes seven calls to console.log to output the following triangle:

#

##

###

####

#####

######

#######

It may be useful to know that you can find the length of a string by writing .length after it.

let abc = "abc";

console.log(abc.length);

// → 3

Solution:

Source Code:

```
let row=0;
let size=7
while (row < size) {
  let col=0;
  var str="";
  while(col<row+1){
        str=str+"#";
     col++;
  }
  console.log(str);
  row++;
}
```

Output:

```
PS G:\programs\node js> node triangle.js
#
##
###
####
#####
######
#######
PS G:\programs\node js>
```

4. Chessboard

Write a program that creates a string that represents an 8×8 grid, using newline characters to separate lines. At each position of the grid there is either a space or a "#" character. The characters should form a chessboard.

Passing this string to console.log should show something like this:

```
 # # # #
# # # #
 # # # #
# # # #
 # # # #
# # # #
 # # # #
# # # #
```

When you have a program that generates this pattern, define a binding size = 8 and change the program so that it works for any size, outputting a grid of the given width and height.

Solution:

Source Code:

```
let row = 0;
let size=8;

while (row < size) {
  let col=0;
  let str="";
  while(col<size){
     if((row+col)%2==0){
```

```
        str=str+" ";
    }else{
        str=str+" #";
    }
    col++;
  }
  console.log(str);
  row++;
}
```

Output:

```
PS G:\programs\node js> node chessBoard.js
  #  #  #  #
 #  #  #  #
  #  #  #  #
 #  #  #  #
  #  #  #  #
 #  #  #  #
  #  #  #  #
 #  #  #  #
PS G:\programs\node js>
```