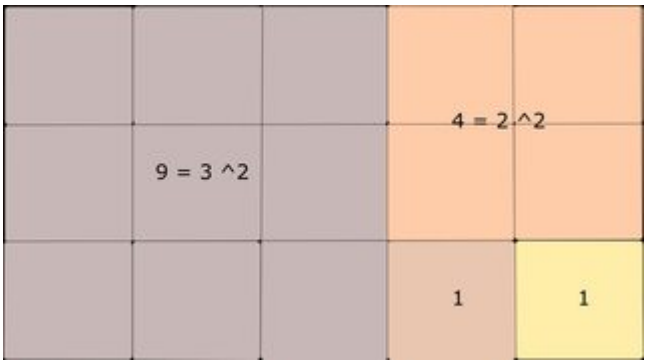
Solve each question in a new tab, do not close tabs!
You are not allowed to search the internet for solutions.

Questions

1. **Minimal Squares in a Rectangle**
   Given a rectangle of dimensions *a* and *b*, we want to divide it into the ***minimal*** number of covering squares. The following rectangle has a dimensions of 5 x 3. It can be cut into 4 squares: [3x3, 2x2, 1x1, 1x1]



   Implement a function that will take two numbers representing the sides of the rectangle and will return an array containing the dimension of each resulting square.

```
var sqrs = squaresInRect(5, 3);
console.log(sqrs);    // [3, 2, 1, 1]

sqrs = squaresInRect(3, 5);
console.log(sqrs);    // [3, 2, 1, 1]

sqrs = squaresInRect(5, 5);
console.log(sqrs);    // [5]
```

2. **Even-Elements-Average**

Write a function that will take an array of numbers, and will return the average of all numbers in the array that are prime number.
Prime number is - a natural number greater than 1 that has no positive divisors other than 1 and itself.

```
function average(array){
...
return ...;
}
var v = [88, 44, 32, 30, 31, 19, 74];
```

```
console.log(average(v)); // 25.0
```

3. Logic Question

Passing the River Five people come to a river at night. There is a narrow bridge, but it can only hold two people at a time. They have one torch and, because it's night, the torch must be used when crossing the bridge. Person A can cross the bridge in one minute, B in 2 minutes, C in 5 minutes, D in 8 minutes and E in 12 minutes. When two people cross the bridge together, they move at the slower person's pace. What is the minimum total time needed for the group to cross? (Give an answer, not a code).

4. **Zero Trim**

   Write a function which trims multiple zero's sequences to a single zero digit. The return value is the updated array. **You are not allowed to use another array**, and you need to implement it with one pass over the array. In other words, each element in the array should change its index only once during the program.

```
var w = [1,2,0,0,0,0,5,7,-6,0,0,0,8,0,0];
var n = zeroTrim(w);
console.log(n);  //print [1,2,0,5,7,-6,0,8,0]
```

   Best implementation should be in complexity of O(n), i.e., with one pass over the array.

5. **Change element to be the following maximal one**

   Write a function which gets and array of numbers and replaces every element with the next greatest element on the right side in the array (don't swap, replace). It replaces the last element with 0 as there no element on the right side of it.

```
var w = [6, 7, 4, 3, 5, 2];
var n = replaceWithNextMax(w);
console.log(n); //print [7, 5, 5, 5, 2, 0]
```

   You should implement it with one pass over the array (usually using one loop only).

   Explanation: for array[0]=6, if we take the rest of the array: {7,4,3,5,2}, the max is "7", so array[0] becomes 7 and the array is now {7,7,4,3,5,2}.
   For array[1]=7, the max from the rest of the array {4,3,5,2} is 5, so array[1] becomes 5, and the array now is {7,5,4,3,5,2}, and so on.

6. **Xbonacci**

Fibonacci series is generated using a simple rule:

$F_n = F_{n-1} + F_{n-2},$

This states that the next element is generated by summing the last two elements.
Fibonacci series is generated by using the first two elements as [1, 1] - this is known as the ***signature*** of the series - and using above rule.
Then, the generated series will be : **1**, **1**, 2, 3, 5, 8, 13, 21

We can generalize this to generate a series from a signature of length N and a rule that generate the next element by summing previous N elements. So if given signature [1,1,1] we generate [1, 1, 1, 3, 5, 9, 17, 31,.....]

Implement a function ***Xbonacci(sig, n)*** that will take a signature array and the length of the series to be generated. It will return a new array of length ***n*** containing the generated series..

```
Xbonacci([1,1], 10)        ===>   [1,1,2,3,5,8,13,21,34,55]
Xbonacci([1,1,1], 8)       ===>   [1,1,1,3,5,9,17,31]
Xbonacci([1,1,1,1], 10)    ===>   [1,1,1,1,4,7,13,25,49,94]
Xbonacci([0,0,0,0,1], 10)  ===>   [0,0,0,0,1,1,2,4,8,16]
```

7. **squeeze**

Write a function **Squeeze(arr1, arr2)** that will take 2 arrays of numbers – arr1 & arr2.
The function should delete each number in arr1 that matches in arr2
The function returns the updated arr1.

```
function Squeeze(arr1, arr2) {
...
return ...;
}
var arr1 = [88, 44, 32, 30, 31, 19, 74];
var arr2 = [32, 19, 74];

console.log(Squeeze(arr1, arr2)); // [88,44,30,31]
```

Good Luck!