```
//PROMISE.ALL:

//O "Promise.all()" é um método especial da função "Promise" que executa uma quantidade infinita de promises e devolve o resultado só depois que todas as promises tiverem terminado suas execuções.

//"Promise.all()" recebe apenas um parâmetro, mas como ele consegue executar várias promises se ele recebe somente 1 parâmetro? Por que essa promises serão passadas dentro de um array, em resultado, "Promise.all()" sempre irá retornar um array de resultados.

//OBS: Quando usamos "Promise.all()" para chamar execuções promises, não precisamos usar "new", afinal as execuções já foram instanciadas elas são promises, o papel de "Promise.all()" é apenas chamar essas instâncias todas de uma vez.

//Vejamos como utilizá-la:

//APLICANDO PROMISE.ALL() NUMA FUNÇÃO GERADORA DE NÚMEROS ALEATÓRIOS:
//Temos abaixo uma função que gerará um número aleatório de determinado número inicial até um final em uma quantidade de tempo pré-determinada:
function numbersRandom(firstPoint, finalPoint, seconds) {
    if(firstPoint > finalPoint) [finalPoint, firstPoint] = [firstPoint, finalPoint] //Faz a validação dos números iniciais e finais...
    return new Promise(resolve => {
        setTimeout(function() {
            const randomNum = parseInt(Math.random() * (finalPoint - firstPoint) + firstPoint) //Gerador de números aleatórios dentro do intervalo passado...
            resolve(randomNum)
        }, seconds) //Aqui a quantidade de tempo é passada...
    })
}

//Abaixo temos uma função que usará "Promise.all()" para gerar vários números aleatórios, note que cada um deles tem um time...
function variousNum() {
    return Promise.all([ //Como todos eles estão dentro de um "Promise.all()" a execução só termina depois que áquele que levar maior tempo - que no caso é 4 segundos - tiver terminado a sua execução, retornando um array com todos os valores...
        numbersRandom(1, 10, 4000),
        numbersRandom(1, 10, 3000),
        numbersRandom(1, 10, 2000),
        numbersRandom(1, 10, 1000),
```

```
        numbersRandom(1, 10, 500),
    ])
}

console.time('promise') //Perceba que aqui temos o marco zero, como essa execução está fora da promise, ela é executada primeiro...

variousNum()
    .then(numbers => console.log(numbers))
    .then(() => { //Perceba que só depois da chamada da promise temos os "console.timeLog" - para mostrar o tempo que levou até o término de todas as execuções - e o "console.timeEnd" que finaliza a execução e além disso mostra o valor final do cronometro...
        console.timeLog('promise'),
        console.timeEnd('promise')
    })
```

RESULTADO NO CONSOLE...

```
[Running] node "c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\156-Promise.all__O_executor_de_multiplas_promises.js"
[ 9, 8, 2, 2, 8 ]
promise: 4.027s
promise: 4.028s

[Done] exited with code=0 in 4.192 seconds
```