# 2821. Delay the Resolution of Each Promise

## Description

Given an array `functions` and a number `ms`, return a new array of functions.

- `functions` is an array of functions that return promises.
- `ms` represents the delay duration in milliseconds. It determines the amount of time to wait before resolving each promise in the new array.

Each function in the new array should return a promise that resolves after a delay of `ms` milliseconds, preserving the order of the original `functions` array. The `delayAll` function should ensure that each promise from `functions` is executed with a delay, forming the new array of functions returning delayed promises.

**Example 1:**

```
 Input:
functions = [
   () => new Promise((resolve) => setTimeout(resolve, 30))
],
ms = 50
 Output: [80]
 Explanation: The promise from the array would have resolved after 30 ms, but it was delayed by 50 ms, thus 30 ms + 50 ms = 80 ms.
```

**Example 2:**

```
 Input:
functions = [
    () => new Promise((resolve) => setTimeout(resolve, 50)),
    () => new Promise((resolve) => setTimeout(resolve, 80))
],
ms = 70
 Output: [120,150]
 Explanation: The promises from the array would have resolved after 50 ms and 80 ms, but they were delayed by 70 ms, thus 50 ms + 70 ms = 120 ms and
80 ms + 70 ms = 150 ms.
```

**Constraints:**

- `functions` is an array of functions that return promises
- `10 <= ms <= 500`
- `1 <= functions.length <= 10`