

Implement Binary Search

Binary search is an $O(\log(n))$ efficiency algorithm for searching a sorted array to find an element. It operates using the following steps:

1. Find the middle `value` of a sorted array. If `value == target` return (found it!).
2. If middle `value < target`, search right half of array in next compare.
3. If middle `value > target`, search left half of array in next compare.

As you can see, you are successively halving an array, which gives you the $\log(n)$ efficiency. For this challenge, we want you to show your work - how you got to the target value... the path you took!

Write a function `binarySearch` that implements the binary search algorithm on an array, returning the path you took (each middle value comparison) to find the target in an array.









The function takes a sorted array of integers and a target value as input. It returns an array containing (in-order) the middle value you found at each halving of the original array until you found the target value. The target value should be the last element of the returned array. If value not is found, return the string `Value Not Found`.

For example, `binarySearch([1,2,3,4,5,6,7],5)` would return `[4,6,5]`.

For this challenge, when halving, you MUST use `Math.floor()` when doing division: `Math.floor(x/2)`. This will give a consistent, testable path.

Note: The following array will be used in tests:

```
const testArray = [
  0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
  19, 20, 21, 22, 23, 49, 70
];
```

	<code>binarySearch</code> should be a function.
	<code>binarySearch(testArray, 0)</code> should return <code>[13, 5, 2, 0]</code> .
	<code>binarySearch(testArray, 1)</code> should return <code>[13, 5, 2, 0, 1]</code> .
	<code>binarySearch(testArray, 2)</code> should return <code>[13, 5, 2]</code> .
	<code>binarySearch(testArray, 6)</code> should return the string <code>Value Not Found</code> .
	<code>binarySearch(testArray, 11)</code> should return <code>[13, 5, 10, 11]</code> .
	<code>binarySearch(testArray, 13)</code> should return <code>[13]</code> .
	<code>binarySearch(testArray, 70)</code> should return <code>[13, 19, 22, 49, 70]</code> .