FIND PEAK ELEMENT

☆ In this problem, we are supposed to find and return the peak element within an array. Peak element is at i^{th} index if $arr[i-1] < arr[i] > arr[i+1]$

An array can have more than 1 peaks. Brute force solution is to take a linear iteration, keep checking for left and right at each step. But optimal solution involves the use of binary search.

We can observe that the array has a sorted tendency, as some parts are sorted. If it only has 1 peak in binary search, the peak can either be at mid, to the right of mid or at the left of mid. We can decide where the peak is w.r.t mid by checking if values before & after mid are increasing or decreasing. We can repeat the Binary Search process along the same rules.

There might be some edge cases with the first and last element as the element, but they can be solved by assuming elements at

-1 and N to be -infinity and modifying our binary search to go from 1 to N-2.

Pseudocode :

```
findPeakElement(arr, N) {
    if (N == 1) return 0
    if (arr[0] > arr[1]) return 0
    if (arr[N-1] > arr[N-2) return N-1
    low = 1
    high = N-2
    while (low <= high) {
        mid = (high + low) / 2
        if (arr[mid] > arr[mid + 1] && arr[mid]
                                    > arr[mid-1]){
            return mid
        } else if (arr[mid] > arr[mid-1]) {
            low = mid + 1
        } else if (arr[mid] > arr[mid + 1]) {
            high = mid - 1
        }
    }
    return -1
}
```

This code will also work seamlessly for more than one peaks.