Medium **Codewriting** **3000**

You have an unsorted array `arr` of non-negative integers and a number `s`. Find a **longest** contiguous subarray in `arr` that has a sum equal to `s`. Return two integers that represent its inclusive bounds. If there are several possible answers, return the one with the smallest left bound. If there are no answers, return `[-1]`.

Your answer should be 1-based, meaning that the first position of the array is `1` instead of `0`.

Example

- For `s = 12` and `arr = [1, 2, 3, 7, 5]`, the output should be `findLongestSubarrayBySum(s, arr) = [2, 4]`.

  The sum of elements from the 2^{nd} position to the 4^{th} position (1-based) is equal to `12`: `2 + 3 + 7`.

- For `s = 15` and `arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`, the output should be `findLongestSubarrayBySum(s, arr) = [1, 5]`.

  The sum of elements from the 1^{st} position to the 5^{th} position (1-based) is equal to `15`: `1 + 2 + 3 + 4 + 5`.

- For `s = 15` and `arr = [1, 2, 3, 4, 5, 0, 0, 0, 6, 7, 8, 9, 10]`, the output should be `findLongestSubarrayBySum(s, arr) = [1, 8]`.

  The sum of elements from the 1^{st} position to the 8^{th} position (1-based) is equal to `15`: `1 + 2 + 3 + 4 + 5 + 0 + 0 + 0`.

Input/Output

- **[execution time limit] 20 seconds (swift)**
- **[input] integer s**

  The sum of the subarray that you are searching for.

  Guaranteed constraints:
  $0 \le s \le 10^9$.

- **[input] array.integer arr**

  The given array.

  Guaranteed constraints:
  $1 \le arr.length \le 10^5$,
  $0 \le arr[i] \le 10^4$.

- **[output] array.integer**

  An array that contains two elements that represent the left and right bounds of the subarray, respectively (1-based). If there is no such subarray, return `[-1]`.

[Swift3] Syntax Tips

```
// Prints help message to the console
// Returns a string
func helloWorld(name: String) -> String {
    print("This prints to the console when you Run Tests");
    return "Hello, " + name;
}
```