1. Código en Go - findPeak

Aquí está el código de Go para la funcion findPeak

```
1
2   package main
3
4   import "fmt"
5
6   // FindPeak finds the peak element in the array
7   func findPeak(arr []int) int {
8     n := len(arr)
9
10    // first or last element is peak element
11    if n == 1 {
12      return 0
13    }
14    if arr[0] >= arr[1] {
15      return 0
16    }
17    if arr[n-1] >= arr[n-2] {
18      return n - 1
19    }
20
21    // check for every other element
22    for i := 1; i < n-1; i++ {
23      // check if the neighbors are smaller
24      if arr[i] >= arr[i-1] && arr[i] >= arr[i+1] {
25        return i
26      }
27    }
28    return -1 // indicating no peak found
29  }
30
31  func main() {
32    arr := []int{1, 3, 20, 4, 1, 0}
33    fmt.Println("Index of a peak point is", findPeak(arr))
34  }
35
```

2. Explicación del código en Go con lstlisting

Aquí está el código traducido a Go junto con una explicación de cada línea:

```
1   package main
2
3   import "fmt"
4
```

Líneas 1-2: Definimos el paquete principal ‘main‘ y luego importamos el paquete ‘fmt‘, que necesitaremos para imprimir resultados en la consola.

```
func findPeak(arr []int) int {
  n := len(arr)
```

Líneas 3-4: Definimos una función llamada ‘findPeak‘ que toma una matriz de enteros como entrada y devuelve un entero (el índice del punto de pico). Calculamos la longitud de la matriz de entrada ‘arr‘ y la almacenamos en la variable ‘n‘.

```
1
2       if n == 1 {
3         return 0
4       }
5
```

Líneas 5-6: Si la longitud de la matriz es 1, significa que hay solo un elemento en la matriz, y ese elemento es el pico. Entonces, devolvemos 0 como el índice del pico.

```
1
2       if arr[0] >= arr[1] {
3         return 0
4       }
5
```

Líneas 7-8: Comprobamos si el primer elemento de la matriz es mayor o igual que el segundo. Si es así, entonces el primer elemento es el pico y devolvemos 0 como el índice del pico.

```
10      if arr[n-1] >= arr[n-2] {
          return n - 1
        }
```

Líneas 9-10: Similar al paso anterior, aquí comprobamos si el último elemento de la matriz es mayor o igual que el segundo desde el final. Si es así, entonces el último elemento es el pico y devolvemos su índice.

```
        for i := 1; i < n-1; i++ {
```

Líneas 11-12: Este bucle itera sobre los elementos de la matriz, excepto el primero y el último.

```
          if arr[i] >= arr[i-1] && arr[i] >= arr[i+1] {
15          return i
          }
        }
        return -1
      }
20
```

Líneas 13-18: Dentro del bucle, comprobamos si el elemento actual es mayor o igual que sus vecinos. Si es así, entonces el elemento actual es el pico y devolvemos su índice. Si no se encuentra ningún pico, devolvemos -1 para indicar que no se encontró ningún pico en la matriz.

```
20
      func main() {
        arr := []int{1, 3, 20, 4, 1, 0}
        fmt.Println("Index of a peak point is", findPeak(arr))
      }
25
```

Líneas 20-22: Ahora definimos la función ‘main‘, que es la entrada principal del programa. Creamos una matriz de enteros llamada ‘arr‘ con valores dados. Llamamos a la función ‘findPeak‘ con la matriz ‘arr‘ y mostramos el índice del punto de pico en la consola.