Given an array of integers [nums] sorted in non-decreasing order, find the starting & ending position of a given target value.

If target is not found in the array, return [-1, -1]

You must write a algorithm with O(log n) runtime complexity

Program

```
#include <stdio.h>
#include <stdlib.h>

int binarySearch(int *nums, int numsSize, int target,
int findFirst){
int left = 0;
int right = numsSize - 1;
int result = -1;

while(left <= right){
  int mid = left + (right - left) / 2;
  if(nums[mid] == target){
    result = mid;
    if(find_first){
      right = mid-1;
      left = mid+1;
    }
    else if(nums[mid] < target){
          left = mid + 1;}

    else {
        right = mid-1;
    }
}
  result = mid-1;
}
```

```
int* SearchRange(int* nums, int numsSize, int target,
int* returnSize) {
int* result = (int*) malloc (2*sizeof(int));
*returnSize = 2;
result[0] = binarySearch(nums, numsSize, target, 1)
result[1] = binary Search(nums, numsSize, target, 0);
return result;
}

int main(){
int nums1[] = {5,7,7,8,8,10};
int target1 = 8;
int size1;

int * result = SearchRange(nums1, 6, target1, &size1);
Printf("output for Example 1:[%d, %d]\n", result[0]
result1[1]);
free(result1);
}
```

output

Input: nums = (5, 7, 7, 8, 8, 10), ~~target~~ 1) target 6

out put : [-1, -1].

8/1/2024