## Problem 1. All subsets of a given Array

Given an integer array of unique elements, return all possible subsets

Subset: A subset of an array is a tuple that can be obtained from the array by removing some (possibly all) elements of it

The solution set must not contain duplicate subsets. Return the solution in any order.

Example 1:

```
Input: array = [1, 2, 3]

Output: [[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]
```

Example 2:

```
Input: array = [0]

Output: [[], [0]]
```

Example 3:

```
Input: array = []

Output: [[]]
```

Constraints:

```
1 <= nums.length <= 10

-10 <= nums[i] <= 10
```

## Problem 2. Find Array Given Subset Sums

You are given an integer n representing the length of an unknown array that you are trying to recover. You are also **given an array containing the values of all $2^n$ subset sums** of **the unknown array** (in no particular order).

Return the array of length n representing the unknown array. If multiple answers exist, return any of them.

An array sub is a subset of an array if sub can be obtained from the array by deleting some (possibly zero or all) elements of the array. The sum of the elements in sub is one possible subset sum of the array. The sum of an empty array is considered to be 0.

Note: Test cases are generated such that there will always be at least one correct answer.

Example 1:

```
Input: n = 3, sums = [-3,-2,-1,0,0,1,2,3]

Output: [1,2,-3]

Explanation: [1,2,-3] is able to achieve the given subset sums:

- []: sum is 0

- [1]: sum is 1

- [2]: sum is 2

- [1,2]: sum is 3

- [-3]: sum is -3

- [1,-3]: sum is -2

- [2,-3]: sum is -1

- [1,2,-3]: sum is 0

Note that any permutation of [1,2,-3] and also any permutation
of [-1,-2,3] will also be accepted.
```

Example 2:

```
Input: n = 2, sums = [0,0,0,0]

Output: [0,0]
```

```
Explanation: The only correct answer is [0,0].
```

Example 3:

```
Input: n = 4, sums = [0,0,5,5,4,-1,4,9,9,-1,4,3,4,8,3,8]
Output: [0,-1,4,5]
Explanation: [0,-1,4,5] is able to achieve the given subset sums.
```

Constraints:

`1 <= n <= 15`

`sums.length ==` $2^n$

`-`$10^{-4}$`<= sums[i] <=` $10^4$

## Problem 3. Count Inversions of an Array

Given an integer array `arr[]` of size `n`, the task is to find the count inversions of the given array. Two array elements `arr[i]` and `arr[j]` form an inversion if `arr[i] > arr[j]` and `i < j`.

Note: Inversion Count for an array indicates how far (or close) the array is from being sorted. If the array is already sorted, then the inversion count is `0`, but if the array is sorted in reverse order, the inversion count is the maximum.

Examples:

**Modify and submit `functions.hpp`. For tests, you can use `main.cpp`.**

Compile command: `g++ -o main main.cpp -std=c++14`

Execution command: `./main "given array"`

Example 1:

```
>> ./main "7 2 6 3"
>> 4
>>
```

Explanation: Given array has 4 inversions: `(7, 2), (7, 6), (7, 3), (6, 3)`

Example 2:

```
>> ./main "1 2 3 4 5"
>> 0
>>
```

Explanation: There is no pair of indexes `i, j` exists in the given array such that

`arr[i] > arr[j]` and `i < j`

## Problem 4. Reverse Pairs

Given an integer array `nums`, return the number of reverse pairs in the array.

A reverse pair is a pair `(i, j)` where:

`0 <= i < j < nums.length` and

`nums[i] > 2 * nums[j]`.

Examples:

**Modify and submit `functions.hpp`. For tests, you can use `main.cpp`.**

Compile command: `g++ -o main main.cpp -std=c++14`

Execution command: `./main "given array"`

Example 1:

```
    >> ./main "1 3 2 3 1"
    >> 4
    >>
```

Explanation:

```
    (1, 4) --> nums[1] = 3, nums[4] = 1, 3 > 2 * 1
    (3, 4) --> nums[3] = 3, nums[4] = 1, 3 > 2 * 1
```

Example 2:

```
    >> ./main "2 4 3 5 1"
    >> 3
    >>
```

Explanation:

```
    (1, 4) --> nums[1] = 4, nums[4] = 1, 4 > 2 * 1
    (2, 4) --> nums[2] = 3, nums[4] = 1, 3 > 2 * 1
    (3, 4) --> nums[3] = 5, nums[4] = 1, 5 > 2 * 1
```

Constraints:

`1 <= nums.length <= 5 * 10`$^4$

`-2`$^{31}$` <= nums[i] <= 2`$^{31}$` - 1`