Neversitup Test

1. **README.md** show your past or standard of init project structure and defined convention
2. Show coding logic and unit test **Permutations** (Choose your the most skilled)

your task is to create all permutations of a non-empty input string and remove duplicates, if present.  
 Create as many **"shufflings"** as you can!

**Examples:**

| **With** input 'a': Your **function** **should** **return**: **['a']**  **With** input 'ab': Your **function** **should** **return** **['ab',** 'ba**']**  **With** input 'abc': Your **function** **should** **return** **['abc','acb','bac','bca','cab','cba']**  **With** input 'aabb': Your **function** **should** **return** **['**aabb**',** 'abab', 'abba', 'baab', 'baba', 'bbaa**']** |
| --- |

1. Show coding logic and unit test **Find the odd int** (Choose your the most skilled)

Given an array of integers, find the one that appears an odd number of times.

**Examples:**

| [7] should return 7, because it occurs 1 time (which is odd). [0] should return 0, because it occurs 1 time (which is odd). [1,1,2] should return 2, because it occurs 1 time (which is odd). [0,1,0,1,0] should return 0, because it occurs 3 times (which is odd). [1,2,2,3,3,3,4,3,3,3,2,2,1] should return 4, because it appears 1 time (which is odd). |
| --- |

1. Show coding logic and unit test **Count the smiley faces!** (Choose your the most skilled)

Given an array (arr) as an argument complete the function **countSmileys** that should return the total number of smiling faces.

Rules for a smiling face:

* Each smiley face must contain a valid pair of eyes. Eyes can be marked as **:** or **;**
* A smiley face can have a nose but it does not have to. Valid characters for a nose are **-** or **~**
* Every smiling face must have a smiling mouth that should be marked with either **)** or **D**

No additional characters are allowed except for those mentioned.

Valid smiley face examples**: :) :D ;-D :~)**

Invalid smiley faces: **;( :> :} :]**

**Examples:**

| **countSmileys([':)', ';(', ';}', ':-D']); // should return 2; countSmileys([';D', ':-(', ':-)', ';~)']); // should return 3; countSmileys([';]', ':[', ';\*', ':$', ';-D']); // should return 1;** |
| --- |

**Note**

In case of an empty array return 0. You will not be tested with invalid input (input will always be an array). Order of the face (eyes, nose, mouth) elements will always be the same.