APT: CustomerStatics

Problem Statement

You will be given a string list `names`, containing a list of customer names extracted from a database. Your task is to report the customers that occur more than once in this list, and the number of occurrences for each of the repeated customers.

Your method should return the report as string list. Each element in this list should be of the form "NAME OCCURS", where NAME is the name of one customer and OCCURS is the number of times his name occurs in `names`. Sort the result in alphabetical order by customer name.

Specification

```
filename: CustomerStatistics.py

def reportDuplicates(names):
    """
    return string list based on parameter
    names, a string list
    """

    # you write code here
```

Constraints

- `names` contains between 1 and 50 elements, inclusive.
- Each element of `names` contains between 1 and 50 characters, inclusive.
- Each element of `names` contains uppercase letters ('A'-'Z') only.

Examples

1. `names = ["JOHN", "BOB", "JOHN", "BILL", "STANLEY", "JOHN"]`

   `Returns: ["JOHN 3"]`

   The only repeated name is JOHN, and it occurs three times.

2. `names = ["YETTI", "YETTI", "YETTI", "BIGFOOT", "BIGFOOT"]`

   `Returns: ["BIGFOOT 2", "YETTI 3" ]`

   Note the sorting order.

3. `names = ["ANDREW", "BILL", "CINDY", "DOH", "ERGH", "FOO", "GOO", "HMPH"]`

   `Returns: []`

   No repeated names this time.

4. `names = ["THEONLYCUSTOMER"]`

   `Returns: []`

   Again, no repeats.

5. `names = ["A", "B", "A", "C", "A", "B", "A", "D", "D", "D"]`

   `Returns: ["A 4", "B 2", "D 3"]`

APT: SerialNumbers

Problem Statement

You own a lot of guitars, and each guitar has a unique serial number. You want to be able to look up serial numbers quickly, so you decide to sort the entire list as follows.

Each serial number consists of uppercase letters ('A' - 'Z') and digits ('0' - '9'). To see if serial number A comes before serial number B, use the following steps:

Specification

```
filename: SerialNumbers.py

def sortSerials(numbers):
    """
    return sorted list of string based on numbers
    a list of strings
    """

    # you write code here
```

1. If A and B have a different length, the one with the shortest length comes first.
2. Else if sum_of_digits(A) differs from sum_of_digits(B) (where sum_of_digits(X) returns the sum of all digits in string X), the one with the lowest sum comes first.
3. Else compare them alphabetically, where digits come before letters.

Given a list of strings `serialNumbers`, return a list of strings with the ordered list of serial numbers in increasing order.

Constraints

- `numbers` will contain between 1 and 50 elements, inclusive.
- Each element of `numbers` will contain between 1 and 50 characters, inclusive.
- `numbers` will only contain uppercase letters ('A' - 'Z') and digits ('0' - '9').
- All elements of `numbers` will be distinct.

Examples

1. `numbers = ["ABCD","145C","A","A910","Z321"]`

   `Returns: {"A", "ABCD", "Z321", "145C", "A910" }`

   The first serial is "A" because it has the shortest length. All others have length 4, but "ABCD" has the lowest sum. Next lowest is "Z321", and finally "A910" comes before "145C" because "A"

comes before the "1" (they both have sum = 10)

2. `numbers = ["Z19", "Z20"]`

   `Returns: {"Z20", "Z19" }`

   1+9 > 2+0, so "Z20" comes first.

3. `numbers= ["34H2BJS6N","PIM12MD7RCOLWW09","PYF1J14TF","FIPJOTEA5"]`

   `Returns: ["FIPJOTEA5", "PYF1J14TF", "34H2BJS6N", "PIM12MD7RCOLWW09" ]`

4. `numbers = ["ABCDE", "BCDEF", "ABCDA", "BAAAA", "ACAAA"]`

   `Returns: ["ABCDA", "ABCDE", "ACAAA", "BAAAA", "BCDEF" ]`

APT: TeamSplit

Problem Statement

Specification

```
filename: TeamSplit.py

def difference(strengths):
    """
    return int based on int list parameter strengths
    """

    # you write code here
```

You want to split some people into two teams to play a game. In order to make the split, you first designate two team captains who take alternate turns selecting players for their teams. During each turn, the captain selects a single player for his team. Since each captain wants to make the strongest possible team, he will always select the best available player. The players have strengths, which are given in int list parameter `strengths`, where higher numbers indicate better players. After all the players have been selected, the team strength is computed as the sum of the strengths of the players on that team.

For example, if `strengths`=[5,7,8,4,2], then the first captain selects the player with strength 8 for his team, the second captain gets the player with strength 7, the first gets the one with strength 5, the second the one with strength 4, and the last one (strength 2) goes to the first team. The first team now has a total strength 8+5+2=15, and the second team has strength 7+4=11.

Return the absolute strength difference between the two teams. For the example above you should return 4 (=15-11).

Constraints

- `strengths` will have between 1 and 50 elements, inclusive.
- Each element of `strengths` will be between 1 and 1000, inclusive.

Examples

1. `strengths = [5,7,8,4,2]`

   `Returns: 4`

   The example from the problem statement.

2. `strengths = [100]`

```
Returns: 100
```

A boring game with only one player. The second team has strength 0 (no players).

3.
```
strengths = [1000,1000]

Returns: 0
```

Both teams with equal strength.

4.
```
strengths = [9,8,7,6]

Returns: 2
```

5.
```
strengths = [1,5,10,1,5,10]

Returns: 0
```

6.
```
strengths = [824, 581, 9, 776, 149, 493, 531, 558, 995, 637,
 394, 526, 986, 548, 344, 509, 319,  37, 790, 491,
 479,  34, 776, 321, 258, 851, 711, 365, 763, 355,
 386, 877, 596,  96, 151, 166, 558, 109, 874, 959,
 845, 181, 976, 335, 930,  22,  78, 120, 907, 584]

Returns: 478
```