- Given a string consisting of 'S' and 'P'
- String must be divided into sections in which exactly two 'S''s exist in each subdivision. The number of 'P''s is irrelevant.
- Return the number of unique subdivisions given a list of 'S' and 'P''s

I love questions like this!

1.) Count the number of 'S' values in input.

2.) Count the number of 'P' values in input.

3.) Base case checks.

  a.) If the count of 'S' is odd or 0, answer is 0.

  b.) If there are zero 'P', or exactly 2 'S' answer is 1

4.) The meat.

Find the second 'S' and third 'S'

Multiply distance between 2nd - 3rd into accumulating answer.

2nd = 3rd→next

3rd = 2nd→next

Solution

```
f ( input )
    want = 'S'
    meh = 'P'
    numWant = count ( input, want)
    numMeh = count ( input, meh)
```

if odd number of want, or non-existent, there is no way to subdivide →

```
    if  isOdd(numWant) or numWant == 0
        return 0
```

There exists 0 locations in which to divide

```
    if  numWant == 2  or  numMeh == 0
        return  1
    ans = 1
    front  =  findSecond( input, vant)
    back  =  find ( front + 1, want)
    while  back
        ans *=  distance( front, back)
        front =  find ( back + 1, want)
        back =  find ( front + 1, want)
    return  ans
```