hw5

hw5

- Reading
- Programming Problems
  - `doubleVowel`
  - `numPairs`
  - `hideShow`
  - `clean`
  - `sequence`

Reading

Chapter 5

Programming Problems

`doubleVowel`

Write a function `doubleVowel` that accepts a word as an argument and returns `True` if the word contains two adjacent vowels and `False` otherwise. Sample usage:

```
>>> doubleVowel('apple')
False
>>> doubleVowel('pear')
True
>>> doubleVowel('peAr')
True
>>> doubleVowel('DURIAN')
True
>>> doubleVowel('baNaNa')
False
>>> doubleVowel('baNaNa')==False
True
```

`numPairs`

Write a function `numPairs` that accepts two arguments, a target number and a list of numbers. The function then **returns** the count of pairs of numbers from the list that sum to the target number. In the first example the answer is 2 because the pairs (0,3) and (1,2) both sum to 3. The pair can be two of the same number, e.g. (2,2) but only if the two 2's are separate twos in the list. In the last example below, there are three 2's, so there are three different pairs (2,2) so there are 5 pairs total that sum to 4. Sample usage:

```
>>> numPairs( 3, [0,1,2,3] )
2
>>> numPairs( 4, [0,1,2,3] )
1
>>> numPairs( 6, [0,1,2,3] )
0
>>> numPairs( 4, [0,1,2,3,4,2] )
3
>>> numPairs( 4, [0,1,2,3,4,2,2] )
5
>>> numPairs( 4, [0,1,2,3,4,2,2] )==5
True
```

`hideShow`

Write a function `hideShow` that accepts two string arguments, an *input* string and a *masking* string. The masking string is a string consisting of '0's and '1's that has the same length as the input string. The function then **returns** a new string that is the same as the input string, except that it is *masked*. That is, in any position where the masking string contains a '0' the input character is replaced by a '#', whereas if the masking string contains a '1', the character is unchanged. Sample usage:

```
>>> hideShow('apple','11001')
'ap##e'
>>> hideShow('apple','00000')
'#####'
>>> hideShow('apple','11111')
'apple'
>>> hideShow('abcdefghijklmnopqrstuvwxyz',13*'01')
'#b#d#f#h#j#l#n#p#r#t#v#x#z'
>>> hideShow( 'df###re##', '101010101' )
'd######e##'
>>> hideShow( 'df###re##', '101010101' )=='d######e##'
True
```

`clean`

Write a function `clean` that when is given a string and returns the string with the leading and trailing space characters removed. Details:

- you must use `while` loop(s)
- you must **not** use the `strip` method
- the space characters are the space `' '`, newline `'\n'`, and tab `'\t'`

```
>>> clean("     hello      ")
'hello'
>>> clean(" hello, how are you?   ")
'hello, how are you?'
>>> clean("\n\n\t   what's up,\n\n doc?  \n \t")
"what's up,\n\n doc?"
>>> clean("\n\n\t   what's up,\n\n doc?  \n \t")=="what's up,\n\n doc?"
True
```

`sequence`

Write a function `sequence` that accepts an number (`int`) and that **prints** a sequence of numbers that starts at the given number and obeys the following rules:

- the number 1 is the last number in the sequence (e.g stop)
- if the number if even, the next number is half of it
- if the number is odd, the next number is one more

Specifications:

- use a `while` loop
- print each number in the sequence one per line

```
>>> sequence(3)
3
4
2
1
>>> sequence(4)
4
2
1
>>> sequence(17)
17
18
9
10
5
6
3
4
2
1
```