1-Bir stringde eğer en az iki kere 'zip' geçiyorsa ikinci kez geçen zip in indexini döndüren, eğer en az iki kere geçmiyorsa -1 döndüren fonksiyon yazalım.

In [3]:

```
def zip_(yazı):
    a = yazı.count("zip")
    if a < 2:
        return -1
    else:
        b = yazı.split("zip")
        return len(b[0] + b[1]) + 3


zip_("abczipabdzipabezip")
```

Out[3]:

```
9
```

In [1]:

```
yazı = "abczipabdzipabezip"
a = yazı.find("zip")
a
```

Out[1]:

```
3
```

In [2]:

```
b = yazı.split("zip")
b
```

Out[2]:

```
['abc', 'abd', 'abe', '']
```

```
2. liste ve boolean alan, "Ping"lerin arasına "Pong" yazan, boolean True ise son
a da koyan False varsa sonuna koymayan bir fonksiyon yazalım
print(yerlestir(["Ping","Ping"],True)) -->['Ping', 'Pong', 'Ping', 'Pong']
print(yerlestir(["Ping","Ping","Ping","Ping","Ping","Ping"],False)) -->['Ping',
 'Pong', 'Ping', 'Pong', 'Ping', 'Pong', 'Ping', 'Pong', 'Ping', 'Pong', 'Ping']
```

In [6]:

```
def yerleştir(liste, bool):
    edited = []
    for i in liste:
        edited.append(i)
        edited.append("Pong")
    if bool:
        return edited
    else:
        del edited[-1]
        return edited



yerleştir(["Ping","Ping","Ping","Ping"],True)
```

Out[6]:

```
['Ping', 'Pong', 'Ping', 'Pong', 'Ping', 'Pong', 'Ping', 'Pong']
```

3. Sayının eklemeli devamlılığını bulan fonksiyon yazalım

eklemeli_devamlılık(1679583)

1+6+7+9+5+8+3=39

3+9=12

1+2=3

Tek basamaklı sayıya ulaşmak için 3 tekrar gerekiyor

In [11]:

```
def eklemeli(x):
    x = str(x)
    a = len(x)
    count = 0
    while a != 1:
        new = 0
        for i in x:
            new += int(i)
        x = str(new)
        a = len(x)
        count += 1
    return count


eklemeli(1679583)
```

Out[11]:

```
3
```

```
4. You will be given an array of numbers. You have to sort the odd numbers in as
cending order while leaving the even numbers at their original positions.

Example:

[7, 1] => [1, 7]

[5, 8, 6, 3, 4] => [3, 8, 6, 5, 4]

[9, 8, 7, 6, 5, 4, 3, 2, 1, 0] => [1, 8, 3, 6, 5, 4, 7, 2, 9, 0]
```

In [13]:

```python
def asc(liste):
    odd =[]
    for i in liste:
        if i % 2:
            odd.append(i)
    odd = sorted(odd)

    result = []
    k = 0
    for i in liste:
        if i % 2:
            result.append(odd[k])
            k += 1
        else:
            result.append(i)
    return result


    #odd = [1,3,5,7,9]

asc([5, 8, 6, 3, 4])
```

Out[13]:

```
[3, 8, 6, 5, 4]
```

```
5- Ornek olarak verilen stringin her bir karakteri sirasiyla buyuk harf haline g
elecek ve sonuc bir liste seklinde olacak. Ornek 1:

test = ‘iki kelime’

sonuc = ['Iki kelime', 'iKi kelime', 'ikI kelime', 'iki Kelime', 'iki kElime',
 'iki keLime', 'iki kelIme', 'iki keliMe', 'iki kelimE']
```

hatalı çözüm

In [26]:

```python
def upper(yazı):
    sonuc = []
    k = 0
    for i in yazı:
        if i.isalnum():
            a = yazı.replace(i,i.upper())
            sonuc.append(a)
            k += 1
    return sonuc


upper("ankara")
```

Out[26]:

```
['AnkArA', 'aNkara', 'anKara', 'AnkArA', 'ankaRa', 'AnkArA']
```

doğru çözüm

In [31]:

```python
def upper(yazı):
    sonuc=[]
    for i in range(len(yazı)):
          if yazı[i].isalpha():
            sonuc.append(yazı[:i]+yazı[i].upper()+yazı[i+1:])
    return sonuc
upper("ankara")
```

Out[31]:

```
['Ankara..', 'aNkara..', 'anKara..', 'ankAra..', 'ankaRa..', 'ankarA..']
```

1. ``` morse_codes = { 'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.', 'G': '--.', 'H': '....', 'I': '..', 'J': '.---', 'K': '-.-', 'L': '.-..', 'M': '--', 'N': '-.', 'O': '---', 'P': '.--.', 'Q': '--.-', 'R': '.-.', 'S': '...', 'T': '-', 'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-', 'Y': '-.--', 'Z': '--..', ' ': ' ', '0': '-----', '1': '.----', '2': '..---', '3': '...--', '4': '....-', '5': '.....', '6': '-....', '7': '--...', '8': '---..', '9': '----.', '&': '.-...', "'": '.----.', '@': '.--.-.', ')': '-.--.-', '(': '-.--.', ':': '---...', ',': '--..--', '=': '-...-', '!': '-.-.--', '.': '.-.-.-', '-': '-....-', '+': '.-.-.', '"': '.-..-.', '?': '..--..', '/': '-..-.' }

``` Arasında boşluk bırakılarak girilen morse kodunu ifadeye, ifadeyi morsea çeviren kod yazalım.
```

In [29]:

```
morse_codes = {
  'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.',
  'G': '--.', 'H': '....', 'I': '..', 'J': '.---', 'K': '-.-', 'L': '.-..',
  'M': '--', 'N': '-.', 'O': '---', 'P': '.--.', 'Q': '--.-', 'R': '.-.',
  'S': '...', 'T': '-', 'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-',
  'Y': '-.--', 'Z': '--..', ' ': ' ', '0': '-----',
  '1': '.----', '2': '..---', '3': '...--', '4': '....-', '5': '.....',
  '6': '-....', '7': '--...', '8': '---..', '9': '----.',
  '&': '.-...', "'": '.----.', '@': '.--.-.', ')': '-.--.-', '(': '-.--.',
  ':': '---...', ',': '--..--', '=': '-...-', '!': '-.-.--', '.': '.-.-.-',
  '-': '-....-', '+': '.-.-.', '"': '.-..-.', '?': '..--..', '/': '-..-.'
}
def dönüştür(yazı):
    a = yazı.split()

    result = ""
    keys = list(morse_codes.keys())
    values = list(morse_codes.values())
    for i in a:
        if i in keys:
           result += morse_codes[i] + " "
        elif i in values:
            result += keys[values.index(i)] + " "
    return result.strip()

#dönüştür("A B C D")
dönüştür("-....- .-.-.")
```

Out[29]:

```
'- +'
```

```
7- Once upon a time, on a way through the old wild mountainous west,…
… a man was given directions to go from one point to another. The directions wer
e "NORTH", "SOUTH", "WEST", "EAST". Clearly "NORTH" and "SOUTH" are opposite, "W
EST" and "EAST" too.

Going to one direction and coming back the opposite direction right away is a ne
edless effort. Since this is the wild west, with dreadfull weather and not much
 water, it's important to save yourself some energy, otherwise you might die of
 thirst!

How I crossed a mountainous desert the smart way.
The directions given to the man are, for example, the following (depending on th
e language):

["NORTH", "SOUTH", "SOUTH", "EAST", "WEST", "NORTH", "WEST"].
or
{ "NORTH", "SOUTH", "SOUTH", "EAST", "WEST", "NORTH", "WEST" };
or
[North, South, South, East, West, North, West]
You can immediatly see that going "NORTH" and immediately "SOUTH" is not reasona
ble, better stay to the same place! So the task is to give to the man a simplifi
ed version of the plan. A better plan in this case is simply:

["WEST"]
or
{ "WEST" }
or
[West]
Other examples:
In ["NORTH", "SOUTH", "EAST", "WEST"], the direction "NORTH" + "SOUTH" is going
 north and coming back right away.

The path becomes ["EAST", "WEST"], now "EAST" and "WEST" annihilate each other,
 therefore, the final result is [] (nil in Clojure).

In ["NORTH", "EAST", "WEST", "SOUTH", "WEST", "WEST"], "NORTH" and "SOUTH" are n
ot directly opposite but they become directly opposite after the reduction of "E
AST" and "WEST" so the whole path is reducible to ["WEST", "WEST"].

Task
Write a function dirReduc which will take an array of strings and returns an arr
ay of strings with the needless directions removed (W<->E or S<->N side by sid
e).

The Haskell version takes a list of directions with data Direction = North | Eas
t | West | South.
The Clojure version returns nil when the path is reduced to nothing.
The Rust version takes a slice of enum Direction {North, East, West, South}.
See more examples in "Sample Tests:"
Notes
```

```
Not all paths can be made simpler. The path ["NORTH", "WEST", "SOUTH", "EAST"] i
s not reducible. "NORTH" and "WEST", "WEST" and "SOUTH", "SOUTH" and "EAST" are
 not directly opposite of each other and can't become such. Hence the result pat
h is itself : ["NORTH", "WEST", "SOUTH", "EAST"].
```

In [35]:

```python
def dirReduc(dir):
    while "NS" in dir or "SN" in dir or "WE" in dir or "EW" in dir:
        dir = dir.replace("NS","").replace("SN","").replace("WE","").replace("EW","")
    if dir == "":
        return 0
    else:
        return dir

dirReduc("NWESNWENNS")
#dirReduc("NN")
```

Out[35]:

```
'NN'
```