344. Reverse String

Reverse String

```
class Solution:
    def reverseString(self, s: List[str]) -> None:
        """
        Do not return anything, modify s in-place instead.
        """
        i = 0
        j = len(s) - 1
        while i < j:
            tmp = s[i]
            s[i] = s[j]
            s[j] = tmp
            i += 1
            j -= 1
```

557. Reverse Words in a String III

Reverse Words in a String III

```
class Solution:
    def reverseWords(self, s: str) -> str:
        a = s.split(" ")
        for i in range(len(a)):
            a[i] = a[i][::-1]
        return " ".join(a)
```

709. To Lower Case

To Lower Case

```
class Solution:
    def toLowerCase(self, str: str) -> str:
        return str.lower()
```

804. Unique Morse Code Words

Unique Morse Code Words

```
class Solution:
    def uniqueMorseRepresentations(self, words: List[str]) -> int:

        ss=[".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",".-..","--
","-.","---",".--.","--.-",".-.","...","-","..-","...-",".--","-..-","-.--","--.."]

        res = set()
        for word in words:
            s = ""
            for char in word:
                s += ss[ord(char) - 97]

            res.add(s)

        return len(res)
```

917. Reverse Only Letters

Reverse Only Letters

```
class Solution:
    def reverseOnlyLetters(self, S: str) -> str:
        l = []
        list_s = list(S) # split each element

        for c in reversed(list_s):
            if c.isalpha(): # if letter, add in list
                l.append(c)

        for i,c in enumerate(list_s):
            if not c.isalpha(): # if not letter, insert element with index
                l.insert(i,c)

        return "".join(l) # transform list to string
```

1108. Defanging an IP Address

Defanging an IP Address

```
class Solution:
    def defangIPaddr(self, address: str) -> str:
        return address.replace(".",'[.]')
```

1221. Split a String in Balanced Strings

Split a String in Balanced Strings

```
class Solution:
    def balancedStringSplit(self, s: str) -> int:
        count=0
        word=0

        for i in range(len(s)):
            if s[i] == 'L':
                count += 1
            else:
                count -=1

            if count == 0 :
                word += 1
        return word
```

1309. Decrypt String from Alphabet to Integer Mapping

Decrypt String from Alphabet to Integer Mapping

```
class Solution:
    def freqAlphabets(self, s: str) -> str:
        dict1={'10#':'j','11#':'k','12#':'l','13#':'m','14#':'n','15#':'o',
                 '16#':'p','17#':'q','18#':'r','19#':'s','20#':'t','21#':'u',
                 '22#':'v','23#':'w','24#':'x','25#':'y','26#':'z'}

        dict2={'1':'a','2':'b','3':'c','4':'d','5':'e','6':'f',
                   '7':'g','8':'h','9':'i'}

        for key,value in dict1.items():
            s=s.replace(key,value)

        for key,value in dict2.items():
            s=s.replace(key,value)

        return s
```

1446. Consecutive Characters

Consecutive Characters

```
class Solution:
    def maxPower(self, s: str) -> int:
        # Temp  = [1] in case of input of length 1 string
        temp = [1]
        arr = list(s)
        count = 1
        for i in range(0,len(arr)-1):
            if(arr[i] == arr[i+1]):
                count+=1
                temp.append(count)
            else:
                count=1
        return(max(temp))
```

1436. Destination City

Destination City

```
class Solution:
    def destCity(self, paths: List[List[str]]) -> str:
        initial = []
        final = []
        for i in paths:
            initial.append(i[0])
            final.append(i[1])
        res = list(set(final) - set(initial))
        return res[0]
```