Kunal Kushwaha DSA Bootcamp - String

▼ *Unfinished -*

925. Long Pressed Name

1859. Sorting the Sentence

1668. Maximum Repeating Substring

1108. Defanging an IP Address

```
class Solution {
    public String defangIPaddr(String address) {

            return address.replace(".","[.]");
    }
}
```

1528. Shuffle String

```
class Solution {
    public String restoreString(String s, int[] indices) {

    char [] ans = new char[indices.length]  ;
         for ( int i =0 ;i<indices.length;i++){
            ans[indices[i]]= s.charAt(i) ;
        }

        return new String(ans);
    }


}
```

1678. Goal Parser Interpretation

```
class Solution {
    public String interpret(String command) {
         return command.replace("()","o").replace("(al)","al");
    }
}
```

1859. Sorting the Sentence

```
class Solution {
    public String sortSentence(String s) {

   String[] arr = s.split(" ");
        String[] sorted = new String[arr.length];

        for (String str : arr) {
            int len = str.length();
            int index = str.charAt(len - 1) - '0';
            sorted[index - 1] = str.substring(0, len - 1);
        }
        return String.join(" ", sorted);
    }
}
```

1662. Check If Two String Arrays are Equivalent

```
class Solution {
    public boolean arrayStringsAreEqual(String[] word1, String[] word2) {
        String sen1="";
        String sen2="";
        for( int i = 0;i<word1.length;i++){
            sen1 = sen1+word1[i] ;
        }
System.out.println(sen1)  ;
          for( int i = 0;i<word2.length;i++){
            sen2= sen2+word2[i] ;
        }
System.out.println(sen2)  ;
        return ( sen1.equals(sen2)) ;
    }
}
```

1704. Determine if String Halves Are Alike

```
class Solution {
    public boolean halvesAreAlike(String s) {
  //split code
String sen1 = s.substring(0,s.length()/2) ;
String sen2 = s.substring(s.length()/2) ;
int count1=0;
int count2=0 ;
for( int i =0; i<s.length()/2;i++){
if (sen1.charAt(i)=='a' || sen1.charAt(i)=='e' || sen1.charAt(i)=='i' || sen1.charAt(i)=='o' || sen1.charAt(i)=='u'|| sen1.charAt(i)=='A' |
   count1++ ;
}
}

for( int i =0; i<s.length()/2;i++){
if (sen2.charAt(i)=='a' || sen2.charAt(i)=='e' || sen2.charAt(i)=='i' || sen2.charAt(i)=='o' || sen2.charAt(i)=='u'|| sen2.charAt(i)=='A' |
   count2++ ;
}
}

return count1==count2  ;
    }
}
```

1309. Decrypt String from Alphabet to Integer Mapping

```
We will use the replace function to replace all the digits and # to alphabet.

Note - It is a must to mention the replace functions for j,k,l....,z before a,b,c,....,i because -

For example, s="10#11#12",
if the code is written in the sequence a ,b ,c ,d ,...,z  then it will print the output "a0#aa#ab" , because it hasn't yet found the replac
```

```
class Solution {
    public String freqAlphabets(String s) {
        return(
            s.replace("10#","j").replace("11#","k").replace("12#","l").replace("13#","m").replace("14#","n").replace("15#","o").replace("16
        ) ;
    }
}
```

1967. Number of Strings That Appear as Substrings in Word

```
class Solution
{
   public int numOfStrings(String[] patterns, String word)
   {
        int count = 0;
        for(String pattern : patterns)
        {
            if(word.contains(pattern)) count++;
        }
        return count;
    }
}
```

657. Robot Return to Origin

Consider a graph and a user is at the center of the graph at (0,0).
We know that if we move above the origin along Y-axis the value at Y-axis increases and if we move below the value along Y-axis decreases (X Remains the same) same applies for X-axis too if we move along left of the center the value of X decreases and if we go right the value of x increases so using that,

if we go above origin then y++
if we go below origin then y--
if we go left side of origin x--
if we go right side of origin x++

```
class Solution {
    public boolean judgeCircle(String moves) {
              int x = 0, y = 0;
        for (char move : moves.toCharArray()) {
            if (move == 'U') y++;
            if (move == 'D') y--;
            if (move == 'L') x--;
            if (move == 'R') x++;
        }
        return x == 0 && y == 0;
    }

}
```

557. Reverse Words in a String III

```
- Firstly , we split the string wherever I find a " " i.e space, and save it in a string array "arr" . So now ,"arr" only contains the word
- Then ,we use the stringbuilder class to reverse each word of the sentence present in "arr" , convert it into string and again save it to
- Lastly , we join the elements of the array using " ".
```

```
class Solution {
    public String reverseWords(String s) {
       String[] arr = s.split(" ") ;

       for(int i =0;i<arr.length;i++){
           StringBuilder sb=new StringBuilder(arr[i]);
           sb.reverse();
           arr[i] = sb.toString() ;
       }

        String res = String.join(" ", arr);
        return res;
    }
}
```

28. Find the Index of the First Occurrence in a String

```
class Solution {
    public int strStr(String haystack, String needle) {

         if(haystack.contains(needle))
             return haystack.indexOf(needle);

        else
        return -1;
    }
}
```

125. Valid Palindrome

```
class Solution {
    public boolean isPalindrome(String s) {
    s = s.toLowerCase();
    s = s.replaceAll("\\s", ""); // remove spaces
    s = s.replaceAll("[^a-zA-Z0-9]", "");  // remove non alphanumeric characters
     int i = 0;
     int j = s.length()-1 ;

     while(i<j){

         if(s.charAt(i)!=s.charAt(j)){
         return false;
         }
        i++;
        j--;
     }
     return true;
    }
}
```

Better code -

```
public class Solution {
    public boolean isPalindrome(String s) {
        // convert Strign s to lowerCase with regex
        String tempString = s.replaceAll("[^A-Za-z0-9]", "").toLowerCase();
        // Reverse the tempString
        String rev = new StringBuffer(tempString).reverse().toString();
        // check tempString to rev String
        return tempString.equals(rev);
    }
}
```

680. Valid Palindrome II

Let's consider on example in order to understand it :-
Input: s = "abcdecba"
Output: true

So, what I can do is create two pointer's & start comparing them from.

- One will start from 0th Index & another will start from last index.
- We'll check, if they are equal then continue checking
- But if they are undequal we can have 2 cases :-
    - Case 1 : Skip e to check whether it's forming an palindrome
    - Case 2 : Skip d to check whether it's forming an plaindrome
- But still if after deleting one character we are not gettong palindrome **return false**
- Otherwise **return true**

```
class Solution {
    public boolean validPalindrome(String s) {
        int i = 0;
        int j = s.length() - 1;

        while(i <= j){
            if(s.charAt(i) == s.charAt(j)){
                i++;
                j--;
            }
            else return isPalindrome(s, i + 1, j) || isPalindrome(s, i, j - 1);
        }
        return true;
    }
    public boolean isPalindrome(String s, int i, int j){
        while(i <= j){
            if(s.charAt(i) == s.charAt(j)){
                i++;
                j--;
            }
            else return false;
        }
        return true;
    }
}
```

1784. Check if Binary String Has at Most One Segment of Ones

1784. Check if Binary String Has at Most One Segment of Ones Leetcode

https://youtu.be/zuAH9ReSczs

```
class Solution {
    public boolean checkOnesSegment(String s) {
```

```
    return !s.contains("01");
    }
}
```

1768. Merge Strings Alternately

```
class Solution {
    public String mergeAlternately(String word1, String word2) {

    String ans="";
     int len=Math.max(word1.length(),word2.length());
     for(int i=0;i<len;i++)
     {
         if(i<word1.length())
           ans+=""+word1.charAt(i);
         if(i<word2.length())
           ans+=""+word2.charAt(i);
     }
     return ans;
    }
    }
```

2000. Reverse Prefix of Word

```
class Solution {
    public String reversePrefix(String word, char ch) {
    System.out.println(word.indexOf(ch)) ;
    int flag= 0 ;   // to check if the word contains the character
    int range = 0 ;

// finding the index at which ch is found.
for ( int i =0 ; i< word.length();i++){
    if (word.charAt(i)==ch){
       flag = 1 ;  // setting flag =1 to confirm the presence of ch in word.
       range = i ;
       break ;
     }
  }
String s="" ;
String suffix ="" ;

// if ch in word
if( flag ==1){
        s = word.substring(0,range+1) ;
        StringBuilder sb  = new StringBuilder(s);
        sb.reverse() ; // reverse the substring
        String prefix = sb.toString() ;
        suffix = word.substring(range+1) ; // remaining part of the word
        return(prefix+suffix); // Adding the reversed string and the suffix
   }

else
return word ;

    }
}
```

13. Roman to Integer

https://leetcode.com/problems/roman-to-integer/description/

```
class Solution {
    public int romanToInt(String s) {
        int value = 0;
        int ans = 0;
        int previous = 0 ;
        for ( int i =s.length()-1 ;i>=0 ; i--){
        switch(s.charAt(i)){
            case 'I' -> value = 1;
            case 'V' -> value = 5;
            case 'X' -> value = 10;
            case 'L' -> value = 50;
            case 'C' -> value = 100;
            case 'D' -> value = 500;
            case 'M' -> value = 1000;
        }

       if( value < previous )
         ans = ans - value ;
       else
        ans =  ans + value ;

        previous = value ;
        }
     return ans ;
    }
}
```

20. Valid Parentheses

```
// first in first out (FIFO) method
    Stack<Character> stack = new Stack<Character>();
    for (int i = 0; i < s.length(); i++) {
        // iterate through the string to add(push) into empty stack with "([{" letters only.
        // if characters are not under "([{", then it will be under ")]}"
        if(s.charAt(i) == '(' || s.charAt(i) == '[' || s.charAt(i) == '{') {
            stack.push(s.charAt(i));
        }
        // if chacter is not "([{", then see if the stack is empty.
        // if the stack is not empty, then
        //      remove and get(pop) the bottom element of stack and compare with other pairs of "([{"
        // if popped element is one of those pairs, move on to next element
        // if not, the order is not correct, so return false; ex) ([)] --> false
        else {
            if(s.charAt(i) == ')') {
                if(stack.isEmpty() || stack.pop() != '(') return false;
            }
            if(s.charAt(i) == ']') {
                if(stack.isEmpty() || stack.pop() != '[') return false;
            }
            if(s.charAt(i) == '}') {
                if(stack.isEmpty() || stack.pop() != '{') return false;
            }
        }
    }
    // if the stack is empty, it means the loop went through without any problem --> True
    // if the stack is not empty, stack will have only one single pair of one of "([{"
    //      ex) string s = "["  --> false;
    return stack.isEmpty();
```

58. Length of Last Word

```
class Solution {
    public int lengthOfLastWord(String s) {
        s = s.trim();
        int lastSpaceIdx = s.lastIndexOf(' ');
        return s.length() - lastSpaceIdx - 1;
    }
}
```