- You are given two floating numbers a and b. You need to output a/b and decimal precision of a/b upto 3 places after decimal point.

Example 1:

Input:

5.43 2.653

Output:

2.046739 2.047

Example 2:

Input:

3.25 2.5

Output:

1.3 1.300

****

Given two strings S1 and S2 as input. Your task is to concatenate two strings and then reverse the string. Finally print the reversed string.

Example 1:

Input: S1 = "Geeks" , S2 = "forGeeks"

Output: "skeeGrofskeeG"

- **to swap first and last digits of a number.**
- **to enter a number and print its reverse.**
- **to find frequency of each digit in a given integer.**
- **to enter a number and print it in Roman letter.**
- **to find HCF (GCD) of two numbers.**

Example 2:

Input: S1 = "Practice" , S2 = "Geeks" **Output:** "skeeGecitcarP" **Explanation:**

Concatenating S1 and S2 to get "PracticeGeeks" then reversing it to "skeegecitcarP".

****

Given a array of N strings, find the longest common prefix among all strings present in the array.

Example 1:
Input:
N = 4
arr[] = {geeksforgeeks, geeks, geek, geezer}
Output: gee

Input:
N = 2
arr[] = {hello, world}
Output: -1

****

Given two strings a and b consisting of lowercase characters. The task is to check whether two given strings are an anagram of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters can be different. For example, act and tac are an anagram of each other.

Input:
a = geeksforgeeks,
b = forgeeksgeeks
Output: YES

****

Given a sorted dictionary of an alien language having N words and k starting alphabets of standard dictionary. Find the order of characters in the alien language.

Input:
N = 5, K = 4
dict = {"baa","abcd","abca","cab","cad"}
Output:
1

****

Given two strings A and B. Find the characters that are not common in the two strings.

Input:
A = geeksforgeeks
B = geeksquiz
Output: fioqruz

****

Given a string S. The task is to find the first repeated character in it. We need to find the character that occurs more than once and whose index of second occurrence is smallest. S contains only lowercase letters.

Input: S="geeksforgeeks"
Output: e
Explanation: 'e' repeats at third position.

****

Given a string str and another string patt. Find the minimum index of the character in str that is also present in patt.

Input:
str = geeksforgeeks
patt = set
Output: 1

****

Given a sentence in the form of a string in uppercase, convert it into its equivalent mobile numeric keypad sequence.

Input:
S = "GFG"
Output: 43334
Explanation: For 'G' press '4' one time. For 'F' press '3' three times.

****

Given two strings S and P. Find the smallest window in the string S consisting of all the characters(including duplicates) of the string P. Return "-1" in case there is no such window present. In case there are multiple such windows of same length, return the one with the least starting index.

Input:
S = "timetopractice"
P = "toc"
Output: toprac
Explanation: "toprac" is the smallest substring in which "toc" can be found.

****

Your task is to implement the function strstr. The function takes two strings as arguments (s,x) and locates the occurrence of the string x in the string s. The function returns and integer denoting the first occurrence of the string x in s (0 based indexing).

Input:
s = GeeksForGeeks, x = Fr
Output: -1
Explanation: Fr is not present in the
string GeeksForGeeks as substring.

****

Given an expression string x. Examine whether the pairs and the orders of “{“,”}”,”(“,”)”,”[“,”]” are correct in exp. For example, the function should return 'true' for exp = “[()]{}{[()()]()}” and 'false' for exp = “[(])”.

Example 1:

Input:

{([])}

Output:

true

****

Given two strings a and b. The task is to find if the string 'b' can be obtained by rotating another string 'a' by exactly 2 places.

Input:

a = amazon

b = azonam

Output: 1

****

Given a string in roman no format (s) your task is to convert it to an integer . Various symbols and their values are given below.

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

Input:

s = V

Output: 5

****

Given a string S. The task is to print all unique permutations of the given string in lexicographically sorted order.

Input: ABC

Output:

ABC ACB BAC BCA CAB CBA

****

Given two strings s and t. Return the minimum number of operations required to convert s to t. The possible operations are permitted:

Insert a character at any position of the string.
Remove any character from the string.
Replace any character from the string with any other character.

Input:
s = "geek", t = "gesek"
Output: 1

****

Given two strings, one is a text string and other is a pattern string. The task is to print the indexes of all the occurences of pattern string in the text string. For printing, Starting Index of a string should be taken as 1.

Input:
S = "batmanandrobinarebat", pat = "bat"
Output: 1 18

****

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

P A H N
A P L S I I G
Y I R
And then read line by line: "PAHNAPLSIIGYIR"

Input: s = "PAYPALISHIRING", numRows = 3

Output: "PAHNAPLSIIGYIR"

****

Given two non-negative integers num1 and num2 represented as strings, return the

product of `num1` and `num2`, also represented as a string.

Example 1:

```
Input: num1 = "2", num2 = "3"
Output: "6"
```

****

Given an input string (`s`) and a pattern (`p`), implement wildcard pattern matching with support for `'?'` and `'*'` where:

- **`'?'` Matches any single character.**
- **`'*'` Matches any sequence of characters (including the empty sequence).**

The matching should cover the entire input string (not partial).

Example 1:

```
Input: s = "aa", p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
```

****

Given an array of strings `words` and a width `maxWidth`, format the text such that each line has exactly `maxWidth` characters and is fully (left and right) justified.

- **A word is defined as a character sequence consisting of non-space characters only.**
- **Each word's length is guaranteed to be greater than `0` and not exceed `maxWidth`.**
- **The input array `words` contains at least one word.**

Example 1:

Input: words = ["This", "is", "an", "example", "of", "text", "justification."], maxWidth = 16

Output:

```
[
  "This    is    an",
  "example  of text",
  "justification.  "
]
```

****

We can scramble a string s to get a string t using the following algorithm:

1. **If the length of the string is 1, stop.**
2. **If the length of the string is > 1, do the following:**
   - **Split the string into two non-empty substrings at a random index, i.e., if the string is `s`, divide it to `x` and `y` where `s = x + y`.**
   - **Randomly decide to swap the two substrings or to keep them in the same order. i.e., after this step, `s` may become `s = x + y` or `s = y + x`.**
   - **Apply step 1 recursively on each of the two substrings `x` and `y`.**

Given two strings `s1` and `s2` of the same length, return `true` if `s2` is a scrambled string of `s1`, otherwise, return `false`.

Example 1:

Input: s1 = "great", s2 = "rgeat"

Output: true

Explanation: One possible scenario applied on s1 is:

"great" --> "gr/eat" // divide at random index.

"gr/eat" --> "gr/eat" // random decision is not to swap the two substrings and keep them in order.

"gr/eat" --> "g/r / e/at" // apply the same algorithm recursively on both substrings.

divide at random index each of them.

"g/r / e/at" --> "r/g / e/at" // random decision was to swap the first substring and to keep the second substring in the same order.

"r/g / e/at" --> "r/g / e/ a/t" // again apply the algorithm recursively, divide "at" to "a/t".

"r/g / e/ a/t" --> "r/g / e/ a/t" // random decision is to keep both substrings in the same order.

The algorithm stops now, and the result string is "rgeat" which is s2.

As one possible scenario led s1 to be scrambled to s2, we return true.

****

A message containing letters from A-Z can be encoded into numbers using the following mapping:

'A' -> "1"
'B' -> "2"
...
'Z' -> "26"
To decode an encoded message, all the digits must be grouped then mapped back into letters using the reverse of the mapping above (there may be multiple ways). For example, "11106" can be mapped into:

"AAJF" with the grouping (1 1 10 6)
"KJF" with the grouping (11 10 6)
Given a string s containing only digits, return *the number of ways to decode it*.

The test cases are generated so that the answer fits in a 32-bit integer.

Example 1:

Input: s = "12"

Output: 2

Explanation: "12" could be decoded as "AB" (1 2) or "L" (12).

****

iven two words, beginWord and endWord, and a dictionary wordList, return *all the shortest transformation sequences from* beginWord *to* endWord*, or an empty list if no such sequence exists. Each sequence should be returned as a list of the words* [beginWord,

s1, s2, ..., sk].

Example 1:

Input: beginWord = "hit", endWord = "cog", wordList =

["hot","dot","dog","lot","log","cog"]

Output: [["hit","hot","dot","dog","cog"],["hit","hot","lot","log","cog"]]

Explanation: There are 2 shortest transformation sequences:

"hit" -> "hot" -> "dot" -> "dog" -> "cog"

"hit" -> "hot" -> "lot" -> "log" -> "cog"

****

Given a string s and a dictionary of strings wordDict, return true if s can be segmented into a space-separated sequence of one or more dictionary words.

Example 1:

Input: s = "leetcode", wordDict = ["leet","code"]

Output: true

Explanation: Return true because "leetcode" can be segmented as "leet code".

****

Given an integer columnNumber, return *its corresponding column title as it appears in an Excel sheet.*

For example:

A -> 1

B -> 2

C -> 3

...

Z -> 26

AA -> 27

```
AB -> 28
...
```

Example 1:

```
Input: columnNumber = 1
Output: "A"
```

Example 2:

```
Input: columnNumber = 28
Output: "AB"
```

Example 3:

```
Input: columnNumber = 701
Output: "ZY"
```

****

Given two strings `s` and `t`, *determine if they are isomorphic*.

Two strings `s` and `t` are isomorphic if the characters in `s` can be replaced to get `t`.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

Example 1:

```
Input: s = "egg", t = "add"
Output: true
```

****

Given a string `s` representing a valid expression, implement a basic calculator to evaluate it, and return *the result of the evaluation*.

Note: You are not allowed to use any built-in function which evaluates strings as

mathematical expressions, such as `eval()`.

Example 1:

Input: s = "1 + 1"

Output: 2

Example 2:

Input: s = " 2-1 + 2 "

Output: 3

****

Design a data structure that supports adding new words and finding if a string matches any previously added string.

Implement the `WordDictionary` class:

- **`WordDictionary()` Initializes the object.**
- **`void addWord(word)` Adds `word` to the data structure, it can be matched later.**
- **`bool search(word)` Returns `true` if there is any string in the data structure that matches `word` or `false` otherwise. `word` may contain dots `'.'` where dots can be matched with any letter.**

Example:

Input

["WordDictionary","addWord","addWord","addWord","search","search","search","search"]

[[],["bad"],["dad"],["mad"],["pad"],["bad"],[".ad"],["b.."]]

Output

[null,null,null,null,false,true,true,true]

Explanation

```
WordDictionary wordDictionary = new WordDictionary();
wordDictionary.addWord("bad");
wordDictionary.addWord("dad");
wordDictionary.addWord("mad");
wordDictionary.search("pad"); // return False
wordDictionary.search("bad"); // return True
wordDictionary.search(".ad"); // return True
wordDictionary.search("b.."); // return True
```

****

Convert a non-negative integer `num` to its English words representation.

Example 1:

```
Input: num = 123
Output: "One Hundred Twenty Three"
```

Example 2:

```
Input: num = 12345
Output: "Twelve Thousand Three Hundred Forty Five"
```

****

you are playing the Bulls and Cows game with your friend.

You write down a secret number and ask your friend to guess what the number is. When your friend makes a guess, you provide a hint with the following info:

- **The number of "bulls", which are digits in the guess that are in the correct position.**
- **The number of "cows", which are digits in the guess that are in your secret number but are located in the wrong position. Specifically, the non-bull digits in the guess that could be rearranged such that they become bulls.**

Given the secret number `secret` and your friend's guess `guess`, return *the hint for your*

friend's guess.

The hint should be formatted as `"xAyB"`, where `x` is the number of bulls and `y` is the number of cows. Note that both `secret` and `guess` may contain duplicate digits.

Example 1:

Input: secret = "1807", guess = "7810"

Output: "1A3B"

Explanation: Bulls are connected with a '|' and cows are underlined:

"1807"

|

"7810"

****

Given a string s represents the serialization of a nested list, implement a parser to deserialize it and return *the deserialized* `NestedInteger`.

Each element is either an integer or a list whose elements may also be integers or other lists.

Example 1:

Input: s = "324"

Output: 324

Explanation: You should return a NestedInteger object which contains a single integer 324.

****