Given a string, reverse all of its characters and return the resulting string.

Ex: Given the following strings...

“Cat”, return “taC”

“The Daily Byte”, return "etyB yliaD ehT”

“civic”, return “civic”

Given a string, return whether or not it forms a palindrome ignoring case and non-alphabetical characters.

Note: a palindrome is a sequence of characters that reads the same forwards and backwards.

Ex: Given the following strings...

"level", return true

"algorithm", return false

"A man, a plan, a canal: Panama.", return true

Given a string representing the sequence of moves a robot vacuum makes, return whether or not it will return to its original position. The string will only contain L, R, U, and D characters, representing left, right, up, and down respectively.

Ex: Given the following strings...

"LR", return true

"URURD", return false

"RUULLDRD", return true

Given a string, return whether or not it uses capitalization correctly. A string correctly uses capitalization if all letters are capitalized, no letters are capitalized, or only the first letter is capitalized.

Ex: Given the following strings...

"USA", return true

"Calvin", return true

"compUter", return false

"coding", return true

Given two binary strings (strings containing only 1s and 0s) return their sum (also as a binary string).

Note: neither binary string will contain leading 0s unless the string itself is 0

Ex: Given the following binary strings...

"100" + "1", return "101"

"11" + "1", return "100"

"1" + "0", return "1"

Given an array of strings, return the longest common prefix that is shared amongst all strings.

Note: you may assume all strings only contain lowercase alphabetical characters.

Ex: Given the following arrays...

["colorado", "color", "cold"], return "col"

["a", "b", "c"], return ""

["spot", "spotty", "spotted"], return "spot"

Given a string and the ability to delete at most one character, return whether or not it can form a palindrome.

Note: a palindrome is a sequence of characters that reads the same forwards and backwards.

Ex: Given the following strings...

"abcba", return true

"foobof", return true (remove the first 'o', the second 'o', or 'b')

"abccab", return false

Given an array of integers, return whether or not two numbers sum to a given target, k.

Note: you may not sum a number with itself.

Ex: Given the following...

[1, 3, 8, 2], k = 10, return true (8 + 2)

[3, 9, 13, 7], k = 8, return false

[4, 2, 6, 5, 2], k = 4, return true (2 + 2)

Given a string representing your stones and another string representing a list of jewels, return the number of stones that you have that are also jewels.

Ex: Given the following jewels and stones...

jewels = "abc", stones = "ac", return 2

jewels = "Af", stones = "AaaddfFf", return 3

jewels = "AYOPD", stones = "ayopd", return 0

Given two strings s and t return whether or not s is an anagram of t.

Note: An anagram is a word formed by reordering the letters of another word.

Ex: Given the following strings...

s = "cat", t = "tac", return true

s = "listen", t = "silent", return true

s = "program", t = "function", return false

Given a string, return the index of its first unique character. If a unique character does not exist, return -1.

Ex: Given the following strings...

"abcabd", return 2

"thedailybyte", return 1

"developer", return 0

You are given two strings, s and t which only consist of lowercase letters. t is generated by shuffling the letters in s as well as potentially adding an additional random character. Return the letter that was randomly added to t if it exists, otherwise, return ’ ‘.

Note: You may assume that at most one additional character can be added to t.

Ex: Given the following strings...

s = "foobar", t = "barfoot", return 't'

s = "ide", t = "idea", return 'a'

s = "coding", t "ingcod", return ''

Given two integer arrays, return their intersection.

Note: the intersection is the set of elements that are common to both arrays.

Ex: Given the following arrays...

nums1 = [2, 4, 4, 2], nums2 = [2, 4], return [2, 4]

nums1 = [1, 2, 3, 3], nums2 = [3, 3], return [3]

nums1 = [2, 4, 6, 8], nums2 = [1, 3, 5, 7], return []

Given two strings representing sentences, return the words that are not common to both strings (i.e. the words that only appear in one of the sentences). You may assume that each sentence is a sequence of words (without punctuation) correctly separated using space characters.

Ex: given the following strings...

sentence1 = "the quick", sentence2 = "brown fox", return ["the", "quick", "brown", "fox"]

sentence1 = "the tortoise beat the haire", sentence2 = "the tortoise lost to the haire", return ["beat", "to", "lost"]

sentence1 = "copper coffee pot", sentence2 = "hot coffee pot", return ["copper", "hot"]