**05 - List in Python**

**Ex. No. : 5.1 Date:11/4/24**

**Register No: 231401041 Name:JAGDEO MK**

**Balanced Array**

Assume that the given string has enough memory. Don't use any extra space(IN-PLACE)

**Sample Input 1**

a2b4c6

**Sample Output 1**

aabbbbcccccc

**program**

**def generate\_repeated\_chars(input\_str): result=[]**

**i = 0**

**while i<len (input\_str): char=input\_str[i] count = 0**

**i += 1**

**while i < len(input\_str) and input\_str[i].isdigit(): count = count\*10+ int(input\_str[i])**

**i += 1**

**result.append(char \* count) return ''.join(result)**

**input\_str1=input() output\_str1=generate\_repeated\_chars(input\_str1) print (output\_str1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Input** | **Expected** | **Got** |  |
|  | a2b4c6 | Aabbbbcccccc | aabbbbcccccc |  |
|  | a12b3d4 | aaaaaaaaaaaabbbdddd | aaaaaaaaaaaabbbdddd |  |

**Ex. No. : 5.2 Date:11/4/24**

**Register No: 231401041 Name:JAGDEO MK**

**Check pair with difference k**

Robert is having 2 strings consist of uppercase & lowercase english letters. Now he want to compare those two strings lexicographically. The letters' case does not matter, that is an uppercase letter is considered equivalent to the corresponding lowercase letter.

**Input**

The first line contains **T**. Then **T** test cases follow.

Each test case contains a two lines contains a string. The strings' lengths range from 1 to 100 inclusive. It is guaranteed that the strings are of the same length and also consist of uppercase and lowercase Latin letters.

**Output**

If the first string is less than the second one, print "-1". If the second string is less than the first one, print "1". If the strings are equal, print "0".

Note that the letters' case is not taken into consideration when the strings are compared.

**Constraints**

**1**≤**T**≤**50**

**String length**≤**100**

**For example:**

|  |  |
| --- | --- |
| **Input** | **Result** |
|  | |
| 3 | 0 |
| aaaa | -1 |
| aaaA | 1 |
| abs |  |
| Abz |  |
| abcdefg |  |
| AbCdEfF |  |

**Program**

**for \_ in range(int(input())): s1=input().lower() s2=input().lower() print((s1 > s2) - (s1 < s2))**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Input** | **Expected** | **Got** |  |
|  | |  |  |  |
|  | 3 | 0 | 0 |  |
| Aaaa | -1 | -1 |
| aaaA | 1 | 1 |
| Abs |  |  |
| Abz |  |  |
| Abcdefg |  |  |
| AbCdEfF |  |  |

**Ex. No. : 5.3 Date:11/4/24**

**Register No: 231401041 Name:JAGDEO MK**

**Count Elements**

Given two Strings s1 and s2, remove all the characters from s1 which is present in s2.

**Constraints**

1<= string length <= 200

**Sample Input 1**

experience enc

**Sample Output 1**

xpri

PROGRAM

s1 = input() s2 = input() result = ""

for char in s1:

if char not in s2: result += char

print(result)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | |  |
|  | **Input** | **Expected** | **Got** |
|  |  |  |  |  |
|  | experience | xpri | xpri |

**Ex. No. : 5.4 Date:11/4/24**

**Register No: 231401041 Name: JAGDEO MK**

**Distinct Elements in an Array**

String should contain only the words are not palindrome.

**Sample Input 1**

Malayalam is my mother tongue

**Sample Output 1**

is my mother tongue

program

def is\_palindrome (word):

return word == word[::-1]

def filter\_non\_palindromic\_words(input\_string): words = input\_string.split()

non\_palindromic\_words = [word for word in words if not is\_palindrome (word)] return ' '.join(non\_palindromic\_words)

input\_string = input().lower()

output\_string = filter\_non\_palindromic\_words (input\_string) print(output\_string)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Expected** | **Got** |  |  |
|  | Malayalam is my mother tongue | is my mother tongue | is my mother tongue |  |

**Ex. No. : 5.4 Date:11/4/24**

**Register No: 231401041 Name:JAGDEO MK**

Question text

Given a string S, which contains several words, print the count C of the words whose length is atleast L. (You can include punctuation marks like comma, full stop also as part of the word length. Space alone must be ignored)

**Input Format:**

The first line contains S. The second line contains L.

**Output Format:**

The first line contains C

**Boundary Conditions:**

2 <= Length of S <= 1000

**Example Input/Output 1:**

Input:

During and after Kenyattas inauguration police elsewhere in the capital, Nairobi, tried to stop the opposition from holding peaceful demonstrations.

5

Output:

13

Explanation:

The words of minimum length 5 are During

after Kenyattas inauguration police elsewhere capital, Nairobi, tried opposition holding peaceful

demonstrations

Program

S = input()

L = int(input()) words = S.split() count = 0

for word in words: if len(word) >= L:

count += 1 print(count)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | |  |
|  | **Input** | **Expected** | **Got** |
|  |  |  |  |  |
|  | During and after Kenyattas inauguration police elsewhere in the capital, Nairobi, tried to stop the opposition from holding peaceful demonstrations.  5 | 13 | 13 |

**Ex. No. : 5.6 Date:11/4/24**

**Register No: 231401041 Name:JAGDEO MK**

**Find the Factor**

Find if a String2 is substring of String1. If it is, return the index of the first occurrence. else return -1.

**Sample Input 1** thistest123string 123

**Sample Output 1**

8

**Program**

**x=input() y=input() z=x.find(y) print(z)**

**output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Input** | **Expected** | **Got** |  |
|  | thistest123string 123 | 8 | 8 |  |

**Ex. No. : 5.7 Date:11/4/24**

**Register No: 231401041 Name:JAGDEO MK**

**Merge List**

Write a program that takes as input a string (sentence), and returns its second word in uppercase.

For example:

If input is “Wipro Technologies Bangalore” the function should return “TECHNOLOGIES” If input is “Hello World” the function should return “WORLD”

If input is “Hello” the program should return “LESS”

NOTE 1: If input is a sentence with less than 2 words, the program should return the word “LESS”.

NOTE 2: The result should have no leading or trailing spaces.

**For example:**

|  |  |
| --- | --- |
| **Input** | **Result** |
| Wipro Technologies Bangalore | TECHNOLOGIES |
| Hello World | WORLD |
| Hello | LESS |
|  |  |

Program

def second\_word\_uppercase(sentence): words = sentence.split()

if len(words) < 2: return "LESS"

else:

return words[1].upper()

sentence = input()

result = second\_word\_uppercase(sentence) print(result)

output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Input** | **Expected** | **Got** |  |
|  | Wipro Technologies Bangalore | TECHNOLOGIES | TECHNOLOGIES |  |
|  | Hello World | WORLD | WORLD |  |
|  |  |  |  |  |
|  | Hello | LESS | LESS |  |

**Ex. No. : 5.8 Date:11/4/24**

**Register No: 231401041 Name:JAGDEO MK**

**Merge Two Sorted Arrays Without Duplication**

Write a python to read a sentence and print its longest word and its length

**For example:**

|  |  |
| --- | --- |
| **Input** | **Result** |
|  | |
| This is a sample text to test | sample 6 |

Program

def longest\_word(sentence): words = sentence.split() max\_length = 0 longest\_word = ""

for word in words:

if len(word) > max\_length: max\_length = len(word) longest\_word = word

return longest\_word, max\_length

sentence = input()

result = longest\_word(sentence)

print( result[0]) print(str(result[1]))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Input** | **Expected** | **Got** |  |
|  | This is a sample text to test | sample 6 | sample 6 |  |
|  | Rajalakshmi Engineering College, approved by AICTE | Rajalakshmi 11 | Rajalakshmi 11 |  |
|  | Cse IT CSBS MCT | CSBS 4 | CSBS 4 |  |

**Ex. No. : 5.9 Date:11/4/24**

**Register No: 231401041 Name:JAGDEO MK**

**Print Element Location**

Two string values S1, S2 are passed as the input. The program must print first N characters present in S1 which are also present in S2.

**Input Format:**

The first line contains S1. The second line contains S2. The third line contains N.

**Output Format:**

The first line contains the N characters present in S1 which are also present in S2.

**Boundary Conditions:**

2 <= N <= 10

2 <= Length of S1, S2 <= 1000

**Example Input/Output 1:**

Input:

abcbde cdefghbb 3

Output:

bcd

**Note:**

b occurs twice in common but must be printed only once.

Program

def extract\_common\_chars(s1, s2, n):

common\_chars = [] for char in s1:

if char in s2 and char not in common\_chars: common\_chars.append(char)

if len(common\_chars) == n: break

return ''.join(common\_chars) # Input

s1 = input().strip() s2 = input().strip()

n = int(input().strip()) # Output

print(extract\_common\_chars(s1, s2, n))

output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Input** | **Expected** | **Got** |  |
|  | | | | |
|  | Abcbde cdefghbb 3 | bcd | bcd |  |

**Ex. No. : 5.10 Date:11/4/24**

**Register No: 231401041 Name:JAGDEO MK**

**Strictly increasing**

Write a program to check if two strings are balanced. For example, strings s1 and s2 are balanced if all the characters in the s1 are present in s2. The character’s position doesn’t matter. If balanced display as "true" ,otherwise "false".

**For example:**

|  |  |
| --- | --- |
| **Input** | **Result** |
|  | |
| Yn PYnative | True |

Program

def check\_balance(s1, s2): s1\_set = set(s1)

s2\_set = set(s2)

if s1\_set.issubset(s2\_set): return True

else:

return False

s1 = input() s2 = input()

result = check\_balance(s1, s2)

if result: print("True")

else:

print("False")

Output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Expected** | **Got** | | |
|  | Yn PYnative | True | True |  |
|  | Ynf PYnative | False | False |  |