**WEEK 5**

**Experiments based on Strings and its Operations**

**Reverse**a string **without affecting special characters**  
 Given a string **S**, containing special characters and all the alphabets, reverse the string without affecting the positions of the special characters.  
**Input:**A&B  
**Output:**B&A  
**Explanation**: As we ignore '&' and  
As we ignore '&' and then reverse, so answer is "B&A".

**For example:**

| **Input** | **Result** |
| --- | --- |
| A&x# | x&A# |

Answer:(penalty regime: 0 %)

s = input()

b = list(s)

i = 0

while i < len(b):

if b[i] == '&':

if i > 0 and i < len(b) - 1:

b[i+1], b[i-1] = b[i-1], b[i+1]

i += 1

print("".join(b))

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | A&B | B&A | B&A |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Flag question

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.

Answer:(penalty regime: 0 %)

def count\_words\_with\_length\_atleast(input\_string, min\_length):

words = input\_string.split()

count = sum(1 for word in words if len(word) >= min\_length)

return count

input\_string = input().strip()

min\_length = int(input().strip())

print(count\_words\_with\_length\_atleast(input\_string, min\_length))

|  | **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 |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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  aaaa  aaaA  abs  Abz  abcdefg  AbCdEfF | 0  -1  1 |

Answer:(penalty regime: 0 %)

def compare\_strings(s1, s2):

s1\_lower = s1.lower()

s2\_lower = s2.lower()

if s1\_lower < s2\_lower:

return -1

elif s1\_lower > s2\_lower:

return 1

else:

return 0

T = int(input().strip())

for \_ in range(T):

string1 = input().strip()

string2 = input().strip()

print(compare\_strings(string1, string2))

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

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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 |

Answer:(penalty regime: 0 %)

def are\_strings\_balanced(s1, s2):

set\_s1 = set(s1)

set\_s2 = set(s2)

return set\_s1.issubset(set\_s2)

s1 = input().strip()

s2 = input().strip()

print(are\_strings\_balanced(s1, s2))

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

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Write a python program to count all letters, digits, and special symbols respectively from a given string

**For example:**

| **Input** | **Result** |
| --- | --- |
| rec@123 | 3  3  1 |

Answer:(penalty regime: 0 %)

def count\_letters\_digits\_special\_symbols(s):

letter\_count = 0

digit\_count = 0

special\_symbol\_count = 0

for char in s:

if char.isalpha():

letter\_count += 1

elif char.isdigit():

digit\_count += 1

else:

special\_symbol\_count += 1

return letter\_count, digit\_count, special\_symbol\_count

input\_string = input().strip()

letter\_count, digit\_count, special\_symbol\_count = count\_letters\_digits\_special\_symbols(input\_string)

print(letter\_count)

print(digit\_count)

print(special\_symbol\_count)

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | rec@123 | 3  3  1 | 3  3  1 |  |
|  | P@#yn26at^&i5ve | 8  3  4 | 8  3  4 |  |
|  | abc@12& | 3  2  2 | 3  2  2 |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **6**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

In this exercise, you will create a program that reads words from the user until the user enters a blank line. After the user enters a blank line your program should display each word entered by the user exactly once. The words should be displayed in the same order that they were first entered. For example, if the user enters:

first

second

first

third

second

then your program should display:

first

second

third

Answer:(penalty regime: 0 %)

def display\_unique\_words():

unique\_words=set()

input\_words=[]

while True:

word=input().strip()

if not word:

break

input\_words.append(word)

unique\_words.add(word)

for word in input\_words:

if word in unique\_words:

print(word)

unique\_words.remove(word)

display\_unique\_words()

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | first  second  first  third  second | first  second  third | first  second  third |  |
|  | rec  cse  it  rec  cse | rec  cse  it | rec  cse  it |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **7**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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 |

Answer:(penalty regime: 0 %)

def get\_second\_word\_in\_uppercase(sentence):

words = sentence.split()

if len(words) >= 2:

return words[1].upper()

else:

return "LESS"

sentence = input().strip()

print(get\_second\_word\_in\_uppercase(sentence))

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

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **8**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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.

Answer:(penalty regime: 0 %)

def common\_characters\_in\_first\_N(S1, S2, N):

common\_chars = ""

chars\_added = 0

seen = set()

for char in S1:

if char in S2 and char not in seen:

common\_chars += char

seen.add(char)

chars\_added += 1

if chars\_added == N:

break

return common\_chars

S1 = input().strip()

S2 = input().strip()

N = int(input().strip())

print(common\_characters\_in\_first\_N(S1, S2, N))

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

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **9**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Given a string S which is of the format USERNAME@DOMAIN.EXTENSION, the program must print the EXTENSION, DOMAIN, USERNAME in the reverse order.

**Input Format:**

The first line contains S.

**Output Format:**

The first line contains EXTENSION.  
The second line contains DOMAIN.  
The third line contains USERNAME.

**Boundary Condition:**

1 <= Length of S <= 100

**Example Input/Output 1:**

Input:

abcd@gmail.com

Output:

com  
gmail  
abcd

**For example:**

| **Input** | **Result** |
| --- | --- |
| arvijayakumar@rajalakshmi.edu.in | edu.in  rajalakshmi  arvijayakumar |

Answer:(penalty regime: 0 %)

S = input()

at = S.index('@')

dot = S.index('.')

l = len(S)

i = 0

i1 = dot + 1

m = ""

i2 = at + 1

m1 = ""

m3 = ""

while i1 < l:

m += S[i1]

i1 += 1

while i2 < dot:

m1 += S[i2]

i2 += 1

while i < at:

m3 += S[i]

i += 1

print(m)

print(m1)

print(m3)

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | abcd@gmail.com | com  gmail  abcd | com  gmail  abcd |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **10**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Consider the below words as key words and check the given input is key word or not.

keywords: {break, case, continue, default, defer, else, for, func, goto, if, map, range, return, struct, type, var}

Input format:

Take string as an input from stdin.

Output format:

Print the word is key word or not.

Example Input:

break

Output:

break is a keyword

Example Input:

IF

Output:

IF is not a keyword

**For example:**

| **Input** | **Result** |
| --- | --- |
| break | break is a keyword |
| IF | IF is not a keyword |

Answer:(penalty regime: 0 %)

keywords = {"break", "case", "continue", "default", "defer", "else", "for", "func", "goto", "if", "map", "range", "return", "struct", "type", "var"}

word = input().strip()

if word in keywords:

print(word, "is a keyword")

else:

print(word, "is not a keyword")

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | break | break is a keyword | break is a keyword |  |
|  | IF | IF is not a keyword | IF is not a keyword |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.