Worksheet: Strings

CS 101 Algorithmic Problem Solving

Fall 2023

Name(s): _____

HU ID *(e.g., xy01042)*: _____

1. Reading Aid

Ali is reading a paragraph written in *Camel case*. This way of writing has the following properties:

- It is a concatenation of one or more words consisting of English letters.
- All letters in the first word are lowercase.
- For each of the subsequent words, the first letter is uppercase and rest of the letters are lowercase.

Given a string in Camel case, Ali wants to figure out how many words it contains. Write a function, `un_camel`, that takes a string `my_string` as a parameter, and returns `num_words`, the number of words contained in s.

Hint: Trying using the string method, `isupper`, which takes no argument and returns `True` if the calling string is an upper case letter, and `False` otherwise.

Constraints

- `isinstance(my_string, str)`
- `1 <= len(my_string)<= 1000`

Interaction

The input comprises a single line containing `s`.

The output must contain a single number denoting the value of `num_words`.

Sample

Input	Output
oneTwoThree	3
aliLikesApplesAndBananas	5

In the first case, `s` contains 3 words: "one", "two", and "three".

In the second case, `s` contains 5 words: "ali", "likes", "apples", "and", and "bananas".

Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
noWayHome	3
farFromHome	3
acrossTheSpiderVerse	4

Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

Input: `my_string`
Output: 1 + the number of upper case letters in `my_string`

After defining the function, make sure to call it and print its result. You may assume that input has already been taken.

```
def un_camel(my_string):
    count = 0
    for c in my_string:
        if c.isupper():
            count += 1
    return count + 1

return un_camel(my_string)
```

Dry Run

In the space below, show a dry run of your pseudocode using any input from the Exercise section.

my_string	c	count
noWayHome		0
	n	0
	o	0
	W	1
	a	0
	y	0
	H	2
	o	0
	m	0
	e	0
return 3		

2. Game of Letters

Ahmed and Bareeha are playing a game with a string, `S`. Each player also has an individual string, `A` belonging to Ahmed and `B` belonging to Bareeha. Both of these are empty in the beginning. The game begins with Ahmed's turn, after which both players take alternate turns.

In their turn, a player picks a single character from S, adds it to their individual string, and deletes the picked character from string S.

The game continues until the string S is empty. Both players receive a point each if the strings A and B are the same when the game ends. Otherwise, none of them receives any points.

Write a function, `can_win`, that takes the string S as a parameter, and returns `True` if there exists a sequence of moves such that the strings A and B are the same at the end of the game, and `False` otherwise.

Hint: Trying using the string method, `count`, which takes a string, `sub`, as argument and returns the number of occurrences of `sub` in the calling string.

Constraints

- `isinstance(S, str)`
- `1 <= len(S) <= 1000`

Interaction

The input comprises a single line containing S.

The output must contain "Yes" or "No", depending on whether `A` and `B` can be equal at the end of the game.

Sample

Input	Output
baba	Yes
cbcba	No

In the first case, `A` and `B` can each end up as "ba".

In the second case, there is no sequence of moves that makes `A` and `B` equal to each other by the end of the game.

Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
pqprqr	Yes
abcd	No
ijjkik	Yes

Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

> Input: `S`
> Output: True if every character in `S` appears an even number of times.

Pseudocode

After defining the function, make sure to call it and print its result. You may assume that input has already been taken.

```
def can_win(S):
    for c in S:
        count = S.count(c)
        if count(c) % 2 != 0:
            return False
    return True

if can_win(S):
    print('Yes')
else:
    print('No')
```

Dry Run

In the space below, show a dry run of your pseudocode using any input from the Exercise section.

S	c	count
pqprqr		
	p	2
	q	2
	p	2
	r	2
	q	2
	r	2
return True		

```
print('Yes')
```

3. Secret Mission

Muzna is on a secret mission and receives encoded communication. The encoding rule is as follows. Any occurrence of `k[string]` in the encoded message, or the `cipher`, is to be decoded as a repetition of `string` exactly `k` times. For example, `3[ac]` stands for `acacac`.

Write a function, `decode`, that takes `cipher` as a parameter, and returns the corresponding decoded message.

Given an encoded message m, print the decoded version for Ali.

Constraints

- `isinstance(cipher, str)`
- `1 <= len(cipher) <= 1000`
- Every `k` in `cipher` satisfies `1 <= k <= 50`.
- Every `string` in `cipher` is guaranteed not to contain any digits.

Interaction

The input comprises a single line containing the string `cipher`.

The output must contain the decoded string.

Sample

Input	Output
3[ac]	acacac
2[abc]3[cd]ef	abcabccdcdcdef

In the first case, the decoded string is `ac` repeated 3 times.

In the second case, the decoded string is a concatenation of `abc` repeated 2 times, `cd` repeated 3 times, and `ef`.

Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
3[a]2[bc]	
ab9[cd]2[ef]g	
2[a]2[b]cd	

Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

> Input: `cipher`
> Output: A copy of `cipher` in which a substring enclosed in `k[ ]` are replaced with `k` repetitions of the substring.

Pseudocode

After defining the function, make sure to call it and print its result. You may assume that input has already been taken.

```
def decode(cipher):
    message = ''        # the decoded message to be returned
    in_string = False   # are we in an encoding of the form k[string]
    string = ''         # the string in the encoding that is to be repeated
    k = ''              # the desired number of repetitions of string
    for c in cipher:
        if c in '1234567890':  # we are reading k. store the number.
            k += c
        elif c == '[':  # string follows. activate flag.
            in_string = True
        elif in_string:  # string reading is active
            if c != ']':  # string has not yet ended. store string
                string += c
            else:  # string has ended.
                # store repetition
                message += string * int(k)
                # reset values for next encoding
                in_string = False
                string = ''
                k = ''
            else:
        else:  # not reading string or k. copy as is
            final += c
    return message

print(decode(cipher))
```

Dry Run

In the space below, show a dry run of your pseudocode using any input from the Exercise section.

cipher	c	encoded	message	to_repeat	num_repeat
2[a]2[b]cd		False	''	''	1'
	'2'	True		''	'2'
	'['				
	'a'			'a'	
	']'	False	'aa'	''	''
	'2'	True		''	'2'
	'['				
	'b'			'b'	
	']'	False	'aabb'	''	''
	'c'		'aabbc'		
	'd'		'aabbcd'		
			return 'aabbcd'		

4. Minimize the Decimal Value

Fatima receives a binary string, `B`, and she wants to minimize its decimal value by changing the order of 1's and 0's in it. A binary string can be minimized by placing all the 1's in the rightmost positions.

However, she is restricted to only using reversing of the substrings. For example, on reversing the substring `B[1:4]` for `B='01001'`, we obtain `00011`.

Help Fatima by writing a function, `num_reverses`, that takes `B` as a parameter, and returns the minimum number of reverse operations required to minimize `B`.

Constraints

- `isinstance(B, str)`
- `B` contains '0' and '1' only
- `1 <= len(B) <= 1000`

Interaction

The input comprises a single line containing the binary string B.

The output must contain a single number denoting the number of minimum reverse operations.

Sample

Input	Output
000	0
010101	2

In the first case, B does not contain any 1s, so it needs no reverse operations.

In the second case, one possible minimal sequence of reversals is `B[1:5]` yielding `001011`, followed by `B[2:4]` to yield 000111. The total number of reverse operations is 2.

Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
1001	
1010	
1101001	

Problem Identification
Briefly explain the underlying problem you identified in the above question that led you to your solution.

Input: `B`
Output: The number of occurrences of `10` in B.

Pseudocode After defining the function, make sure to call it and print its result. You may assume that input has already been taken.

```
def num_reverses(B):
    count = B.count('10')
    return count

print(num_reverses(B))
```

Dry Run
In the space below, show a dry run of your pseudocode using any input from the Exercise section.

B	count
1101001	2
return 2	

5. Encryption

Sania is using an encryption algorithm that ensures the isomorphism of the encrypted and decrypted strings. She receives a string `t` and decrypts it to get `s`, but she still needs to make sure that this message has not been altered by hackers during transmission.

Write a function, `are_isomorphic`, that takes strings, `s` and `t` as parameters and returns `True` if the message has been altered, i.e. the strings are not isomorphic, and `False` otherwise.

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. A character may map to itself.

Constraints

- `isinstance(s, str) and isinstance(t, str)`
- `len(s) == len(t) and 1 <= len(t) <= 1000`
- If a letter repeats in `s` or `t`, it does so in a position adjacent to its previous occurrence.

Interaction

The input comprises a single line containing 2 space-separated string s and t respectively.

The output must contain a Yes or No, to state if the the message has been altered or not.

Sample

Input	Output
foo bar	Yes
egg add	No

In the first case, the first `o` in `s` maps to `a` in `t` while the second `o` in `s` maps to `r` in `t`. The strings are not isomorphic and the message has been altered.

In the second case, `e` and `g` in `s` consistently map to `a` and `d` in `t`. The strings are isomorphic and the message has not been altered.

Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
paper title	
pizza olive	
missed croods	

Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

Input: `s, t`
Output: `True` if every character in `s` always maps to the same character in `t`, `False` otherwise.

Pseudocode After defining the function, make sure to call it and print its result. You may assume that input has already been taken.

```
def are_isomorphic(s, t):
    last_s = s[0]
    last_t = t[0]
    for i in range(len(t)):
        this_s = s[i]
        this_t = t[i]
        if this_s == last_s and this_t != last_t:
            return True
        last_s = this_s
        last_t = this_t
    return False

if are_isomorphic(s, t):
    print('Yes')
else:
    print('No')
```

Dry Run

In the space below, show a dry run of your pseudocode using any input from the Exercise section. In the space below, show a dry run of your pseudocode using any input from the Exercise section.

B	count
1101001	2
return 2	

LET'S LEARN TO DEBUG

6. Numbers and Order

Friends of Arsalan regularly play a game. They have blocks each denoting some integer from 0 to 9. These are arranged together in a random manner without seeing to form different numbers keeping in mind that the first block is never a 0. Once they form a number they

read in the reverse order to check if the number and its reverse is the same. If both are same then the player wins.

Arsalan happens to see this game and wants to simulate the same in the computer. Given a sequence of numbers as a string `s`, he wants to check if a player wins or loses.

Constraints

- `isinstance(s, str)`
- `1 <= len(s) <= 1000`
- `s` only contains the digits 0 through 9.
- $1 \leq len(s) \leq 100$

Interaction

The input comprises a single line containing the string `s`.

The output must be "wins" or "loses", indicating whether the player wins or loses respectively.

Sample

Input	Output
343	wins
8898	loses

In the first case, reverse of 343 is 343 as well.

In the second case, reverse is 8988 which is not the same as 8898.

Proposed Solution

```
def numberGame(s):
  index = 0
  for i in s:
    if i != s[index + len(s)]:
      return "loses"
      break
    index += 1
  return "wins"
print(numberGame(s))
```

Dry Run

In the space below, show a dry run of your pseudocode using any input from the Exercise section.

Error Identification

Briefly explain the errors you identified in the proposed code solution. Mention the line

number and errors in each line.

Correct Solution
Rewrite the lines of code you mentioned above with their errors corrected.

Rough Work