Write a program for syntax checking of English sentence-english grammar

A simple sentence if syntactically correct if it fulfills given rules. The following are given rules.

1. Sentence must start with a Uppercase character (e.g. Noun/ I/ We/ He etc.)
2. Then lowercase character follows.
3. There must be spaces between words.
4. Then the sentence must end with a full stop(.) after a word.
5. Two continuous spaces are not allowed.
6. Two continuous upper case characters are not allowed. However, the sentence can end after an upper case character.

# Python program to validate a given sentence for a set of rules

# Method to check a given sentence for given rules

def checkSentence(string):

# Calculate the length of the string.

length = len(string)

# Check that the first character lies in [A-Z].

# Otherwise return false.

if string[0] < 'A' or string[0] > 'Z':

return False

# If the last character is not a full stop(.) no

# need to check further.

if string[length-1] != '.':

return False

# Maintain 2 states. Previous and current state based

# on which vertex state you are. Initialise both with

# 0 = start state.

prev\_state = 0

curr\_state = 0

# Keep the index to the next character in the string.

index = 1

# Loop to go over the string.

while (string[index]):

# Set states according to the input characters in the

# string and the rule defined in the description.

# If current character is [A-Z]. Set current state as 0.

if string[index] >= 'A' and string[index] <= 'Z':

curr\_state = 0

# If current character is a space. Set current state as 1.

else if string[index] == ' ':

curr\_state = 1

# If current character is a space. Set current state as 2.

else if string[index] >= 'a' and string[index] <= 'z':

curr\_state = 2

# If current character is a space. Set current state as 3.

else if string[index] == '.':

curr\_state = 3

# Validates all current state with previous state for the

# rules in the description of the problem.

if prev\_state == curr\_state and curr\_state != 2:

return False

# If we have reached last state and previous state is not 1,

# then check next character. If next character is '\0', then

# return true, else false

if prev\_state == 2 and curr\_state == 0:

return False

# Set previous state as current state before going over

# to the next character.

if curr\_state == 3 and prev\_state != 1:

return True

index += 1

prev\_state = curr\_state

return False

# Driver program

string = ["I love cinema.", "The vertex is S.",

"I am single.", "My name is KG.",

"I lovE cinema.", "GeeksQuiz. is a quiz site.",

"I love Geeksquiz and Geeksforgeeks.",

" You are my friend.", "I love cinema"]

string\_size = len(string)

for i in range(string\_size):

if checkSentence(string[i]):

print ("\"" + string[i] + "\" is correct")

else:

print ("\"" + string[i] + "\" is incorrect")

# This code is contributed by BHAVYA JAIN

C program

// C program to validate a given sentence for a set of rules

#include<stdio.h>

#include<string.h>

#include<stdbool.h>

// Method to check a given sentence for given rules

bool checkSentence(char str[])

{

// Calculate the length of the string.

int len = strlen(str);

// Check that the first character lies in [A-Z].

// Otherwise return false.

if (str[0] < 'A' || str[0] > 'Z')

return false;

//If the last character is not a full stop(.) no

//need to check further.

if (str[len - 1] != '.')

return false;

// Maintain 2 states. Previous and current state based

// on which vertex state you are. Initialise both with

// 0 = start state.

int prev\_state = 0, curr\_state = 0;

//Keep the index to the next character in the string.

int index = 1;

//Loop to go over the string.

while (str[index])

{

// Set states according to the input characters in the

// string and the rule defined in the description.

// If current character is [A-Z]. Set current state as 0.

if (str[index] >= 'A' && str[index] <= 'Z')

curr\_state = 0;

// If current character is a space. Set current state as 1.

else if (str[index] == ' ')

curr\_state = 1;

// If current character is [a-z]. Set current state as 2.

else if (str[index] >= 'a' && str[index] <= 'z')

curr\_state = 2;

// If current state is a dot(.). Set current state as 3.

else if (str[index] == '.')

curr\_state = 3;

// Validates all current state with previous state for the

// rules in the description of the problem.

if (prev\_state == curr\_state && curr\_state != 2)

return false;

if (prev\_state == 2 && curr\_state == 0)

return false;

// If we have reached last state and previous state is not 1,

// then check next character. If next character is '\0', then

// return true, else false

if (curr\_state == 3 && prev\_state != 1)

return (str[index + 1] == '\0');

index++;

// Set previous state as current state before going over

// to the next character.

prev\_state = curr\_state;

}

return false;

}

// Driver program

int main()

{

char \*str[] = { "I love cinema.", "The vertex is S.",

"I am single.", "My name is KG.",

"I lovE cinema.", "GeeksQuiz. is a quiz site.",

"I love Geeksquiz and Geeksforgeeks.",

" You are my friend.", "I love cinema" };

int str\_size = sizeof(str) / sizeof(str[0]);

int i = 0;

for (i = 0; i < str\_size; i++)

checkSentence(str[i])? printf("\"%s\" is correct \n", str[i]):

printf("\"%s\" is incorrect \n", str[i]);

return 0;

}