1. Задание

Дан текст на английском языке. Необходимо закодировать его с помощью азбуки Морзе, где каждой букве соответствует последовательность точек и тире. Например, буква «g» превратится в строку «--.». В переменной morze для удобства хранится словарь соответствия латинских букв коду Морзе.

morze = {'a': '.-', 'b': '-…', 'c': '-.-.', 'd': '-..',

'e': '.', 'f': '..-.', 'g': '--.', 'h': '….',

'i': '..', 'j': '.---', 'k': '-.-', 'l': '.-..',

'm': '--', 'n': '-.', 'o': '---', 'p': '.--.',

'q': '--.-', 'r': '.-.', 's': '…', 't': '-',

'u': '..-', 'v': '…-', 'w': '.--', 'x': '-..-',

'y': '-.--', 'z': '--..'}

**На входе:** В одной строке вам дан текст, который состоит из латинских букв и пробелов.

**На выходе:**

Выведите каждое слово исходного текста, закодированное азбукой Морзе. Количество строк в ответе должно совпадать с количеством слов в исходном тексте. Между закодированными буквами ставится ровно один пробел. Например, слово «Help» превратится в «.... . .-.. .--.». Строчные и заглавные буквы кодируются одинаково.

Например:

Ignition sequence start

Перевод

.. --. -. .. - .. --- -.

… . --.- ..- . -. -.-. .

… - .- .-. –

Приложение 1 - Код программы

morze = {'a': '.-', 'b': '-…', 'c': '-.-.', 'd': '-..',  
 'e': '.', 'f': '..-.', 'g': '--.', 'h': '….',  
 'i': '..', 'j': '.---', 'k': '-.-', 'l': '.-..',  
 'm': '--', 'n': '-.', 'o': '---', 'p': '.--.',  
 'q': '--.-', 'r': '.-.', 's': '…', 't': '-',  
 'u': '..-', 'v': '…-', 'w': '.--', 'x': '-..-',  
 'y': '-.--', 'z': '--..'}  
  
inp = input("Введите текст: ")  
inp = inp.lower()  
translate\_text = ""  
for lett in inp:  
 if lett != " ":  
 translate\_text += morze[lett]  
 else:  
 translate\_text += "\n"  
print(translate\_text)

1. Задание

В некотором городе открывается новая служба по доставке электронных писем. Необходимо наладить систему регистрации новых пользователей.

Регистрация должна работать следующим образом: если новый пользователь хочет зарегистрироваться на сайте, то он должен послать системе запрос name со своим именем. Система должна определить, существует ли уже такое имя в базе данных. Если такого имени не существует, то оно заносится в базу данных системы и пользователю возвращается ответ "ОК", подтверждающий успешную регистрацию. А если пользователь с таким именем уже существует, то система должна сформировать новое имя и выдать его пользователю в качестве подсказки, при этом сама подсказка также добавляется в базу данных. Новое имя формируется следующим образом: к name последовательно приписываются числа, начиная с 1 (name1, name2 и так далее), и среди них находят такое наименьшее i, что namei еще не содержится в системе.

**Входные данные**

В первой строке входных данных задано число n (1 ≤ n ≤ 100000). Следующие n строк содержат запросы к системе. Каждый запрос представляет собой непустую строку длиной не более 32 символов, состоящую только из строчных букв латинского алфавита.

**Выходные данные**

В выходных данных должно содержаться n строк – ответы системы на запросы: "OK" в случае успешной регистрации, или подсказка с новым именем, если запрашиваемое уже занято.

Приложение 2 – Код программы

def check(names, new\_nick):  
 nick = str(input())  
 if nick in names:  
 names[nick] += 1  
 nick += str(names[nick])  
 new\_nick.append(nick)  
 else:  
 names[nick] = 0  
 new\_nick.append("OK")  
 return (new\_nick)  
  
  
def main():  
 names = {}  
 new\_nick = []  
 n = input()  
 for i in range(int(n)):  
 check(names, new\_nick)  
  
 for name in new\_nick:  
 print(name)  
  
 return  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

1. Задание

Необходимо создать программу обработки запросов пользователей к файловой системе компьютера. Над каждым файлом можно производить следующие действия: запись – w ("write"), чтение – r ("read"), запуск – x ("execute").

**Входные данные**

На вход программе подаются следующие параметры: число n – количество файлов в файловой системе. В следующих n строках содержится информация с именами файлов и допустимыми действиями (w, x, r), разделенных пробелами. Далее идет число m – количество запросов к файлам вида «операция файл» (обозначение операции: "write", "read", "execute").

**Выходные данные**

Для каждого допустимого запроса программа должна возвращать OK, для недопустимого – Access denied.

Приложение 3 – Код программы

def create\_files(files):  
 up = str(input())  
 up = up.split()  
 files[up[0]] = up[1:]  
 return(files)  
  
def read\_files(files, request):  
 action = str(input())  
 command = action.split()[0]  
 file\_name = action.split()[1]  
 match command:  
 case "read":  
 command = "r"  
 case "write":  
 command = "w"  
 case "execute":  
 command = "x"  
 case \_:  
 print("wrong command!")  
 return  
 if command in files[file\_name]:  
 request.append("OK")  
 else:  
 request.append("Access denied")  
 return(request)  
  
  
  
def main():  
 files = {}  
 request = []  
 n = input()  
 for i in range(int(n)):  
 create\_files(files)  
 m = input()  
 for k in range(int(m)):  
 read\_files(files,request)  
 for record in request:  
 print(record)  
 return 0  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()