MORSE\_CODE\_DICT = { 'A':'2', 'B':'22',

'C':'222', 'D':'3', 'E':'33',

'F':'333', 'G':'4', 'H':'44',

'I':'444', 'J':'5', 'K':'55',

'L':'555', 'M':'6', 'N':'66',

'O':'666', 'P':'7', 'Q':'77',

'R':'777', 'S':'7777', 'T':'8',

'U':'88', 'V':'888', 'W':'9',

'X':'99', 'Y':'999', 'Z':'9999'}

# Function to encrypt the string

# according to the morse code chart

def encrypt(message):

cipher = ''

for letter in message:

if letter != ' ':

# Looks up the dictionary and adds the

# correspponding morse code

# along with a space to separate

# morse codes for different characters

cipher += MORSE\_CODE\_DICT[letter] + '0'

else:

# 1 space indicates different characters

# and 2 indicates different words

cipher += '0'

return cipher

# Function to decrypt the string

# from morse to english

def decrypt(message):

# extra space added at the end to access the

# last morse code

message += ' '

decipher = ''

citext = ''

for letter in message:

if(letter=='0'):

letter=' '

# checks for space

if (letter != ' '):

# counter to keep track of space

i = 0

# storing morse code of a single character

citext += letter

# in case of space

else:

# if i = 1 that indicates a new character

i += 1

# if i = 2 that indicates a new word

if i == 2 :

# adding space to separate words

decipher += ' '

else:

# accessing the keys using their values (reverse of encryption)

decipher += list(MORSE\_CODE\_DICT.keys())[list(MORSE\_CODE\_DICT

.values()).index(citext)]

citext = ''

return decipher

# Hard-coded driver function to run the program

def main():

message = input()

result = decrypt(message)

print (result)

# Executes the main function

if \_\_name\_\_ == '\_\_main\_\_':

main()