```
"""
Rashad Khan
010713326
CS2520-01
Project 2
03/08/2023
"""
import time
#task 1 import

import turtle
#task 2 import


#Task 1

def get_num_of_charecters(str):
    length = len(str)
    for i in str:
        if i == ' ':
            length -=1
    return length
#only spaces are not considered charecters

def output_without_whitespace(str):
    global newstr
    newstr = ''
    for i in str:
        if i == ' ':
            continue
        newstr = newstr + i
    return newstr
#does not concatinate the space litteral

def get_safe(str):
    encrypted = ""
    for i in str:
        # shift the character by 3 on the ASCII chart
        encrypting = chr((ord(i) + 3) % 128)
        encrypted += encrypting
    return encrypted

def go_recover(encrypted):
    decrypted = ""
    for i in encrypted:
        # shift the character back by 3 on the ASCII chart
        message = chr((ord(i) - 3) % 128)
        decrypted += message
    return decrypted


test= input("enter a random phrase: ")
enc= get_safe(test)
print("This is the encrypted message: ",enc)
dec= go_recover(enc)
print("You entered: ",dec)
print("Number of charcters: ",get_num_of_charecters(test))
print("String with no whitespace: ",output_without_whitespace(test))

"""
enter a random phrase: The only thing we have to fear is fear itself.
This is the encrypted message:  Wkh#rqo|#wklqj#zh#kdyh#wr#ihdu#lv#ihdu#lwvhoi1
You entered:  The only thing we have to fear is fear itself.
Number of charcters:  37
```

```
String with no whitespace:  Theonlythingwehavetofearisfearitself.
"""

#task 2
def primes():
    """Generate an infinite sequence of prime numbers."""
    num = 2
    while True:
        is_prime = True
        for i in range(2,num):
            if num % i == 0:
                is_prime = False
                break
        if is_prime:
            yield  num
        num += 1

start_time = time.time()

primeGen=primes()
for i in range(1,1001):
    if i <= 50 or i == 101 or i ==1000:
        prime = next(primeGen)
        print("prime number ",i," :",prime)
    next(primeGen)

end_time = time.time()

print("Time taken:", end_time - start_time, "seconds")

"""
prime number  1  : 2
prime number  2  : 5
prime number  3  : 11
prime number  4  : 17
prime number  5  : 23
prime number  6  : 31
prime number  7  : 41
prime number  8  : 47
prime number  9  : 59
prime number  10  : 67
prime number  11  : 73
prime number  12  : 83
prime number  13  : 97
prime number  14  : 103
prime number  15  : 109
prime number  16  : 127
prime number  17  : 137
prime number  18  : 149
prime number  19  : 157
prime number  20  : 167
prime number  21  : 179
prime number  22  : 191
prime number  23  : 197
prime number  24  : 211
prime number  25  : 227
prime number  26  : 233
prime number  27  : 241
prime number  28  : 257
prime number  29  : 269
prime number  30  : 277
prime number  31  : 283
prime number  32  : 307
prime number  33  : 313
prime number  34  : 331
```

```
prime number  35  : 347
prime number  36  : 353
prime number  37  : 367
prime number  38  : 379
prime number  39  : 389
prime number  40  : 401
prime number  41  : 419
prime number  42  : 431
prime number  43  : 439
prime number  44  : 449
prime number  45  : 461
prime number  46  : 467
prime number  47  : 487
prime number  48  : 499
prime number  49  : 509
prime number  50  : 523
prime number  101  : 877
prime number  1000  : 8389
Time taken: 0.24574780464172363 seconds
"""



#task 3


def convert_to_barcode(digzip):
    encoded_zip = "1"
    #add start
    for i in  digzip+"2":
    #+2 because the  check digit is 2
        match i:
            case  "0":
                encoded_zip+= "11000"
            case  "1":
                encoded_zip+= "00011"
            case  "2":
                encoded_zip+= "00101"
            case  "3":
                encoded_zip+= "00110"
            case  "4":
                encoded_zip+= "01001"
            case  "5":
                encoded_zip+= "01010"
            case  "6":
                encoded_zip+= "01100"
            case  "7":
                encoded_zip+= "10001"
            case  "8":
                encoded_zip+= "10010"
            case  "9":
                encoded_zip+= "10100"
            case _ :
                continue
            #default case is so  that no symbols  will be  taken for the barcode
    encoded_zip += "1"
    #add end
    return encoded_zip



def binary_to_turtle(str_binary_zip):
    for i in str_binary_zip:
        turtle.left(90)
        if i == "1":
```

```
            turtle.forward(50)
            turtle.back(50)
        #if 1 then long line
        else:
            turtle.forward(25)
            turtle.back(25)
        #if  0 (or anything) the  .5 long line
        turtle.right(90)
        turtle.penup()
        turtle.forward(10)
        turtle.pendown()
        #move pen to next spot for barcode
    turtle.hideturtle()
    turtle.done()

turtle.pensize(4)
turtle.speed('fastest')
#turtle  settings

binary_to_turtle(convert_to_barcode(input("Please enter your zip code : ")))
#binary_to_turtle takes a string in binary format
#convert_to_barcodemakes input into a binary string


#screenshots of turtle submitted  seperately
"""
Please enter your zip code : 55555-1237
Please enter your zip code : 91768-1111
Please enter your zip code : 928001-1212
"""
```