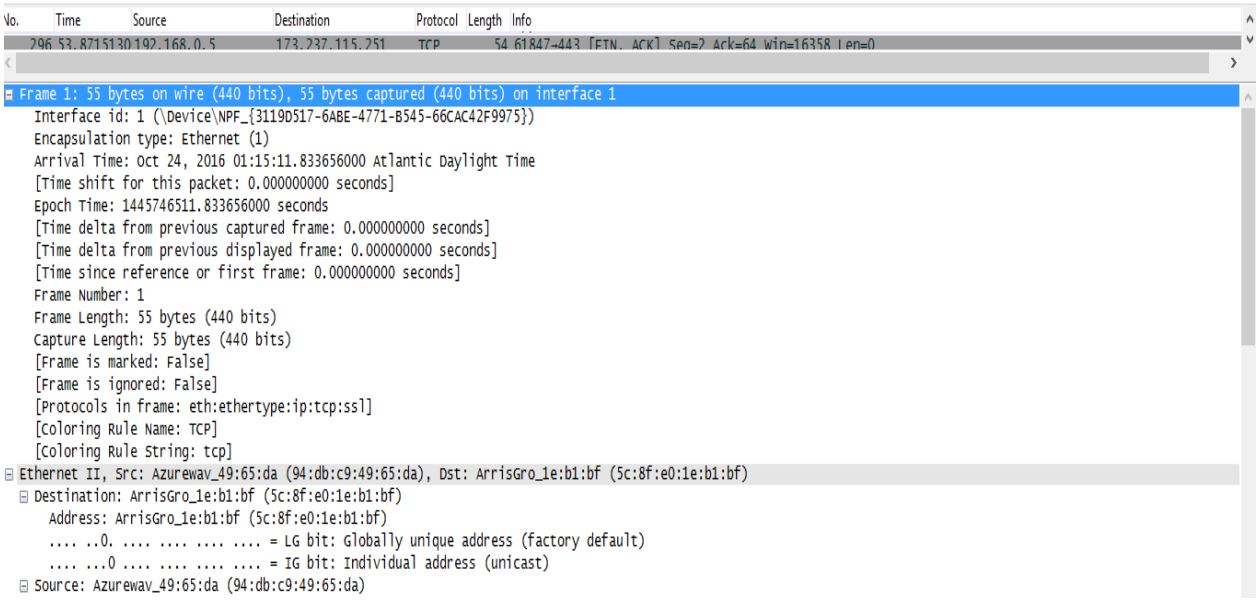
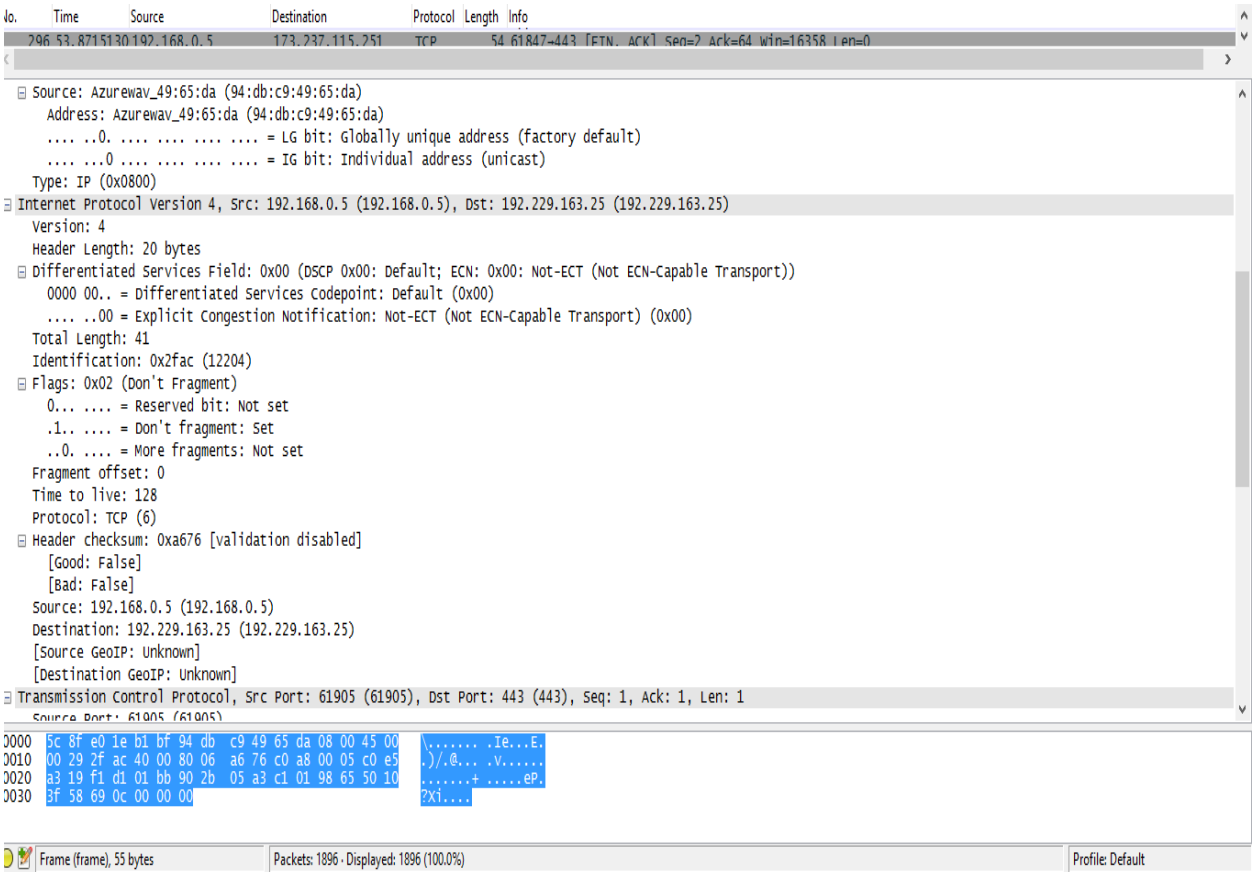
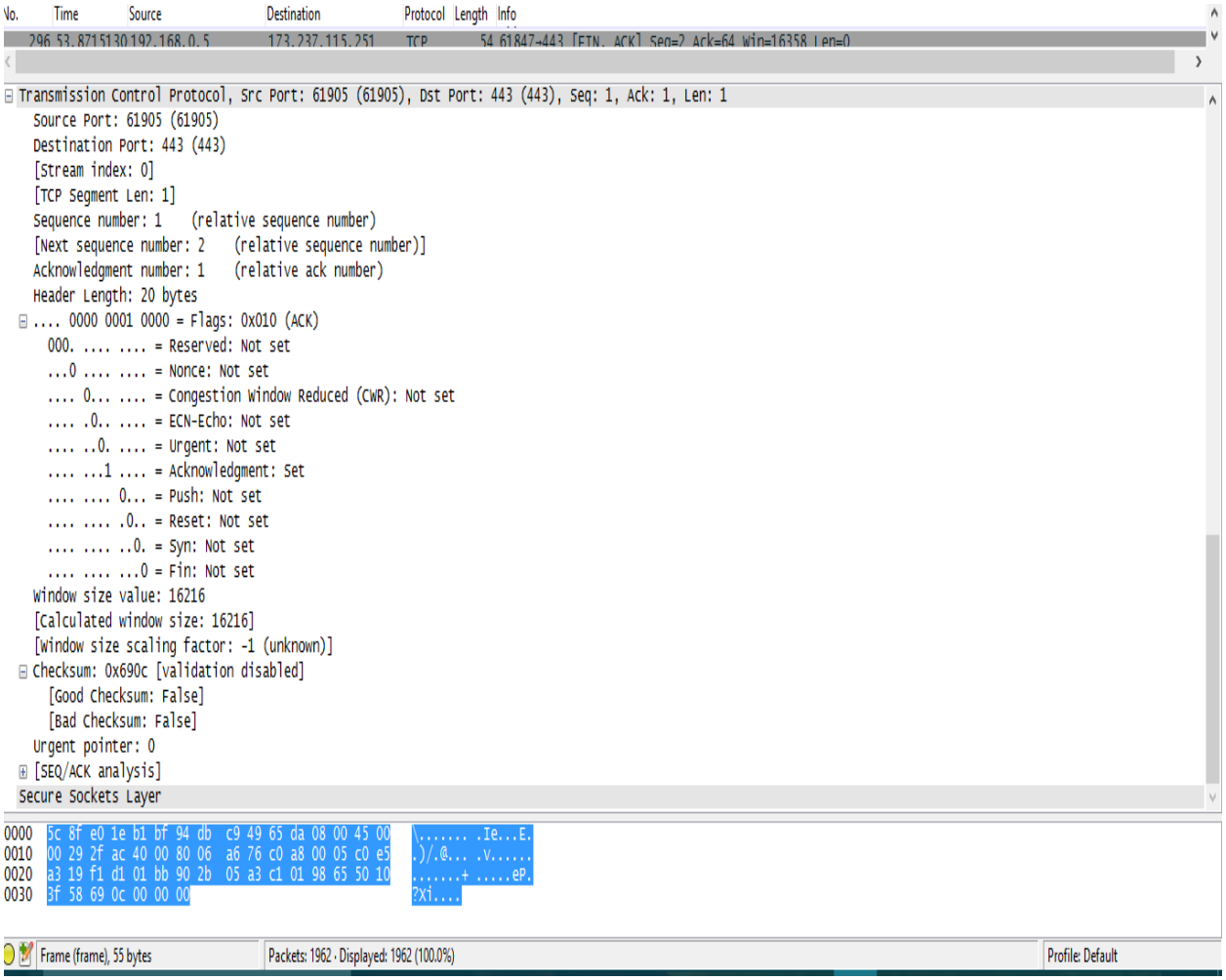
Ans 1.



1. From the screenshot above, type of protocol in this frame can be described. Eg. TCP, IP etc.
2. Frame length can also be determined from this screenshot. And moreover, the frame no. time difference between earlier & latest packet received can be determined.
3. Source and destination address can also be determined.
4. Echo time for the current packet can be determined.



1. Using this screenshot, source address, IP version with source IP, destination IP, header length can be determined.
2. Header checksum info can also be seen in this.
3. Flag info related to fragmentation, offset TTL & protocol can also be determined using this screenshot.

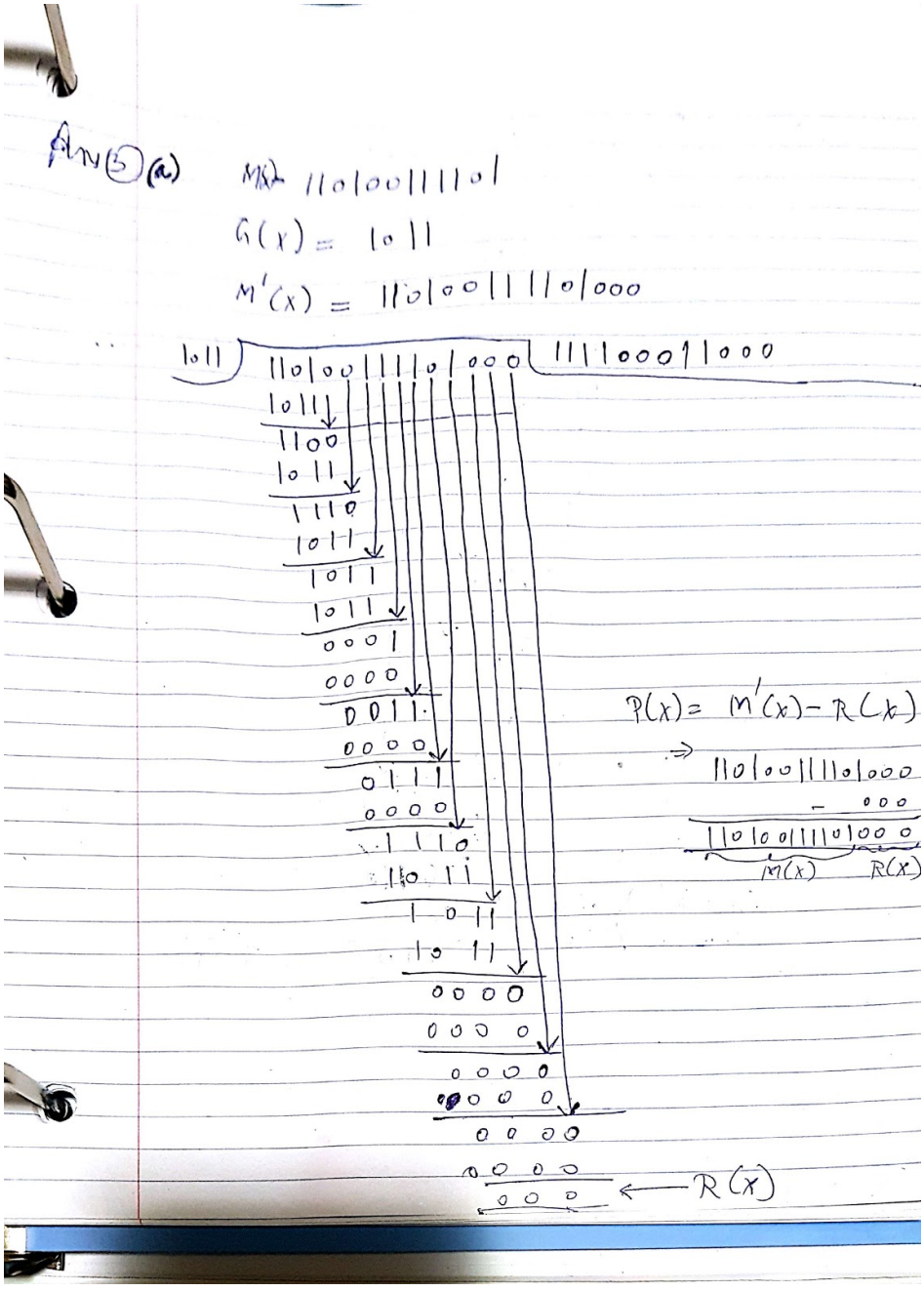


1. Source and destination port numbers, Sequence no., Acknowledgement no., TCP segment no., header length can be determined using this screenshot.
2. The various values on flags can also be determined like Push, Reset, Syn, Finetc, Urgent, Acknowledgement.

Ans2. executable file named Source code.py is attached.
And along with this other files bit_stuffed.txt and data_restored.txt is also attached.

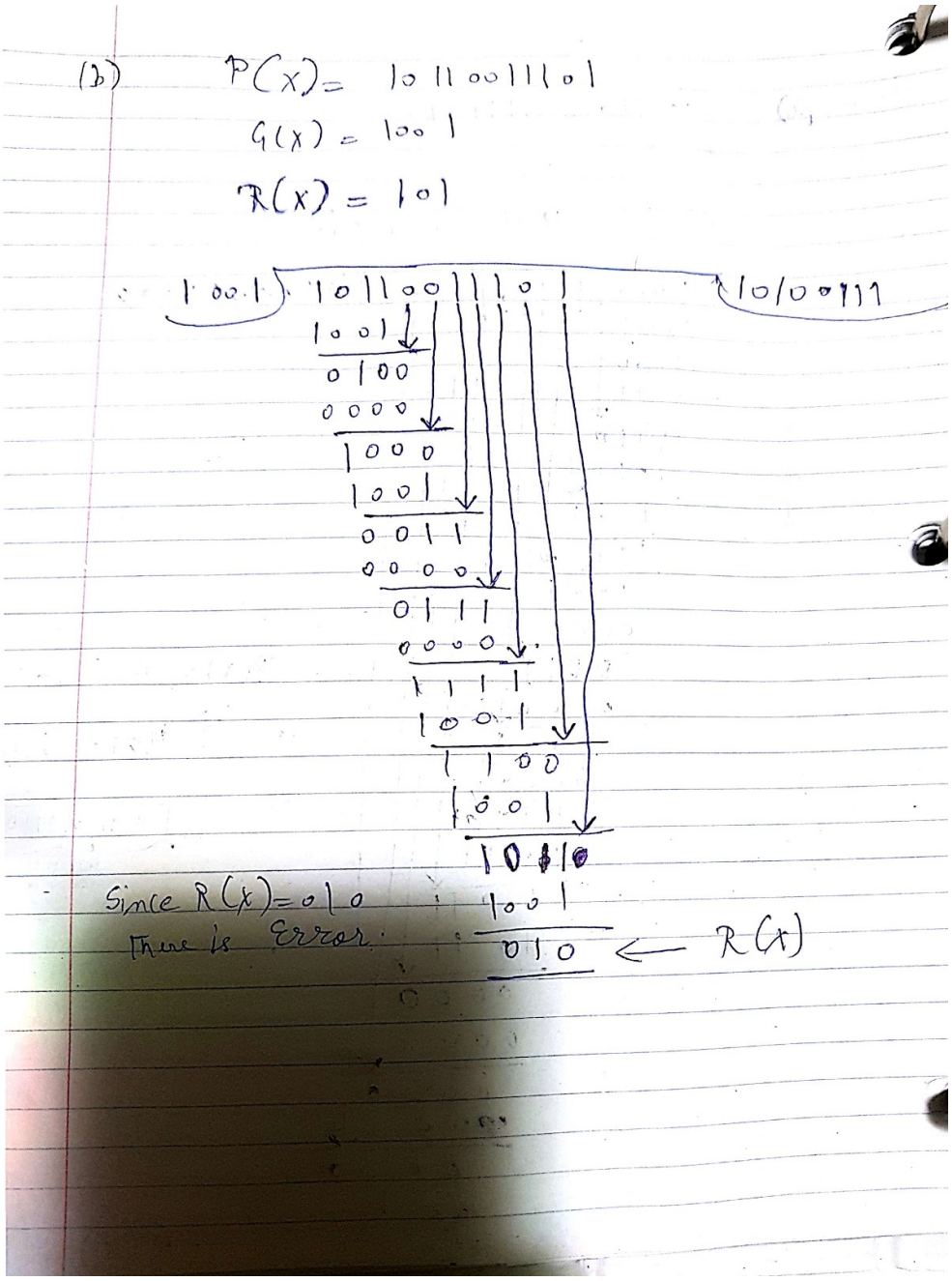
```
import binascii
import re
import sys

def write_file(input_file_path, output_file_path):
   with open("orig_data.txt") as f, open(output_file_path, 'w') as fout:
        for line in f:
            if line.strip():
                binary_string = bin(int(binascii.hexlify(line.strip()), 16))
                bit_stuffed = re.sub('11111', '111110', binary_string)
                fout.write(bit_stuffed+'\n')
            else:
                fout.write(line+'\n')


def read_file(input_file_path, output_file_path):
   with open("bit_stuffed.txt", 'rb') as f, open(output_file_path, 'w') as
fout:
        orig = ''
        for line in f:
            if line.strip():
                bit_unstuffed = re.sub('111110', '11111', line.strip())
                n = int(bit_unstuffed, 2)
                orig = binascii.unhexlify('%x' % n)
                fout.write(orig)
            else:
                fout.write(line)

def main():
    write_file(sys.argv[0], "bit_stuffed.txt")
    read_file("bit_stuffed.txt", "data_restored.txt")

if __name__ == "__main__":
    main()
```

Ans3a



Remainder R(x) is 000 and when this is subtracted from M'(x), we get actual transmitted string that is:- P(x) = M'(x) – R(x) = 1101001111101000
& no error is detected for CRC.

Ans3b

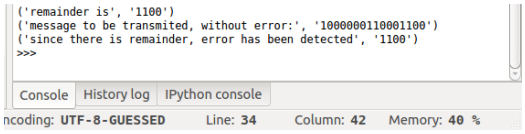


Since we get the remainder R'(x)=010 so "Error has been detected" in the transmitted string.

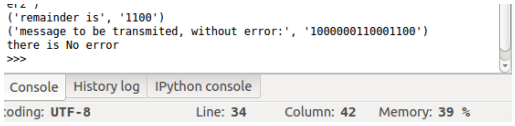
Ans 3c File named Ans3c.py is attached. Code for both a and b are done together below.

```
def transmitter(msg,gen,gencode):
    msgg = msg+gencode
    msg = list(msg)
    msgg = list(msgg)
    gen = list(gen)
    for p in range(len(msgg)-len(gen)):
        if msgg[p] == '1':
            for q in range (len(gen)):
                msgg[p+q] = str((int(msgg[p+q])+int(gen[q]))%2)

    remainder = (msgg[-(len(gen)-1):])
    print("remainder is","".join(remainder))
    transmitted_msg = msg+remainder
    print("message to be transmited, without error:","".join(transmitted_msg))

def error_checker(transmitted_msgg,gen):
    transmitted_msgg = list(transmitted_msgg)
    gen = list(gen)
    for i in range(len(transmitted_msgg)-len(gen)):
        if transmitted_msgg[i] == '1':
            for j in range (len(gen)):
                transmitted_msgg[i+j] = str((int(transmitted_msgg[i+j])+int(gen[j]))%2)

    error = (transmitted_msgg[-(len(gen)-1):])
    error = list(error)
    for m in range(len(error)):
        if error[m] == '1':
            print("since there is remainder, error has been detected","".join(error))
            break
        elif m == (len(error)-1):
            print("there is No error")

transmitter("100000011000","10011","0000")
error_checker("1000000110000000","10011")
```

O/P screenshots:-
transmitter("100000011000","10011","0000")
error_checker("1000000110000000","10011")



If receiver receives the same message as sent from transmitter, then there is no error.
transmitter("100000011000","10011","0000")
error_checker("1000000110001100","10011")



Ans3d
executable file is named Ans3d.py is attached.

Burst error length	No of frame	No of frame error detected
<32 bits	1000	1000
=32 bits	1000	1000
>32 bits	1000	1000

Currently the code is running for error bit less than 32, need to add/remove few comment of code manually to run it under different conditions.

```
import random

x=0
def transmitter(msg,gen,gencode):
    msgg = msg+gencode
    msg = list(msg)
    errorr = "10010001010101111111111000011111"
    errorr = list(errorr)
    gen = list(gen)
    msgg = list(msgg)
    for p in range(len(msgg)-len(gen)):
        if msgg[p] == '1':
            for q in range (len(gen)):
                msgg[p+q] = str((int(msgg[p+q])+int(gen[q]))%2)

    remainder = (msgg[-(len(gen)-1):])
    print("remainder is:","".join(remainder))
    transmitted_msg = msg+remainder
    print("message to be transmitted, without error is:","".join(transmitted_msg))
    #32-bit error_generator, remove comments from below 2 lines to generate 32-bit error and
comment other lines of code accordingly.
    #error_length = 32
    #transmitted_msg = msg+errorr
    #<32-bit error_generator, remove comments from below 3 lines to generate error less than
32-bit and comment other lines of code accordingly.
    for b in range(1505,1533):
        error_length = 27
        transmitted_msg[b] = str(random.randint(0,1))
    #>32-bit_error generator, remove comments from below 3 lines to generate error more than
32-bit and comment other lines of code accordingly.
    '''for b in range(1475,1533):
        error_len = 57
        transmitted_msg[b] = str(random.randint(0,1))'''
    print("message transmitted, with error of length:",error_length,"".join(transmitted_msg))
    error_checker(transmitted_msg,gen)

def error_checker(transmitted_msgg,gen):
    transmitted_msgg = list(transmitted_msgg)
    gen = list(gen)
```

```
    for p in range(len(transmitted_msgg)-len(gen)):
        if transmitted_msgg[p]== '1':
            for q in range (len(gen)):
                transmitted_msgg[p+q] = str((int(transmitted_msgg[p+q])+int(gen[q]))%2)

    error = (transmitted_msgg[-(len(gen)-1):])
    error = list(error)
    for m in range(len(error)):
         if error[m] == '1':
            global x
            x+=1
            print("since there is remainder, error has been detected","".join(error),"& it is in the frame
no.",x)
            break
         elif m == (len(error)-1):
            print("there is No error")

for q in range(1000):
    rand_byte = []
    for p in range(1520):
        d = str(random.randint(0,1))
        rand_byte.append(d)

    rand_bit = "".join(rand_byte)
    new_gen = "10000010011000001000111111O110001"
    new_gen_code = "0"*32

transmitter(rand_bit,new_gen,new_gen_code)
```