PART 1

```
# these are just large numbers to initialize 32bits
bitstring_one = (1 << 31) | 888475480
bitstring_two = (1 << 31) | 1109000000
# print("{:32b}".format(bitstring_one))
# print("{:32b}\n".format(bitstring_two))
output = []
for i in range(64):
  output.append((bitstring_one ^ bitstring_two) & 1)
  # print(f"{(bitstring_two >> 1) & 1} XOR {(bitstring_two >> 10) & 1}")
  bitstring_one_newbit = (bitstring_one ^ (bitstring_one >> 1) ^ (bitstring_one >> 10)) & 1
  # print(f"{(bitstring_two >> 7) & 1} XOR {(bitstring_two >> 13) & 1} XOR {(bitstring_two >> 30)
& 1}")
  bitstring_two_newbit = (bitstring_two ^ (bitstring_two >> 7) ^ (bitstring_two >> 13) ^
(bitstring_two >> 30)) & 1

  # assign new bit to position 31
  bitstring_one = (bitstring_one >> 1) | (bitstring_one_newbit << 31)
  bitstring_two = (bitstring_two >> 1) | (bitstring_two_newbit << 31)

# print("\n{:32b} ".format(bitstring_one))
# print("{:32b} ".format(bitstring_two))
print()
for bit in output[0:20]:
  print(bit, end='')
print("\n")
```

PART 2

```
import re
import random as rd

outputs = []
for i in range(5):
  # these are just large numbers to initialize 32bits
  bitstring_one = (1 << 31) | rd.getrandbits(32)
  bitstring_two = (1 << 31) | rd.getrandbits(32)
  # print("{:32b}".format(bitstring_one))
  # print("{:32b}\n".format(bitstring_two))
  curr_output = []
  for i in range(10 ** 4):
```

```
        curr_output.append((bitstring_one ^ bitstring_two) & 1)
        # print(f"{(bitstring_two >> 1) & 1} XOR {(bitstring_two >> 10) & 1}")
        bitstring_one_newbit = (bitstring_one ^ (bitstring_one >> 2) ^ (bitstring_one >> 3)) & 1
        # print(f"{(bitstring_two >> 7) & 1} XOR {(bitstring_two >> 13) & 1} XOR {(bitstring_two
>> 30) & 1}")
        bitstring_two_newbit = (bitstring_two ^ (bitstring_two >> 1) ^ (bitstring_two >> 2) ^
(bitstring_two >> 3)) & 1

        # assign new bit to position 31
        bitstring_one = (bitstring_one >> 1) | (bitstring_one_newbit << 31)
        bitstring_two = (bitstring_two >> 1) | (bitstring_two_newbit << 31)

  # print("\n{:32b} ".format(bitstring_one))
  # print("{:32b} ".format(bitstring_two))

  outputs.append(curr_output)

output_strings = []
for output in outputs:
  output_strings.append(''.join(map(str,output)))


zero_one_occurences = [] # [0 count(0s), 1 is count(1s)]*len(output_strings)
for output in output_strings:
  curr_zero_one_occurences = []
  curr_zero_one_occurences.append(output.count('0'))
  curr_zero_one_occurences.append(output.count('1'))

  zero_one_occurences.append(curr_zero_one_occurences)
print(zero_one_occurences)

longest_count_occurences = [] # [0 longest(0s), 1 is longest(1s)]*len(output_strings)
for output in output_strings:
  long_zero = 1
  long_one = 1
  curr_zero = 0
  curr_one = 0
  for char in output:
        if(char == '0'):
        curr_zero+=1
        if curr_zero > long_zero: long_zero=curr_zero
        curr_one = 0
        if(char == '1'):
        curr_one+=1
```

```
            if curr_one > long_one: long_one=curr_one
            curr_zero = 0

    longest_count_occurences.append([long_zero, long_one])
print(longest_count_occurences)

sequences = [] # [0 is i=4, 1 is i=5, 2 is i=6]*len(output_strings)
for output in output_strings:
    curr_sequences = []
    curr_sequences.append(len(re.findall("100001",output)))
    curr_sequences.append(len(re.findall("1000001",output)))
    curr_sequences.append(len(re.findall("10000001",output)))

    sequences.append(curr_sequences)
print(sequences)
```