Report for Section 3

Question 5 (A):

1. Start by making for loop run for 1 million times and store the modulo 2 of the randomly generated numbers in an array.
2. Store the first 127 output in a separate array and generate the rest of the array using A[i] = A[i-1] ^ A[i-127].
3. Sum the number of 1s and 0s for both the arrays and output the solution on the terminal

Sample Output:

for rand() modulo 2 the probability and frequency are:
0 => 500249 therefore probability is 0.500313
1 => 499623 therefore probability is 0.499687
for XOR Generator the probability and frequency are:
0 => 499943 therefore probability is 0.500007
1 => 499929 therefore probability is 0.499993

Conclusion:
No matter the way we make the array, the distribution remains to be even.

Question 5 (B):

1. Start by making for loop run for 1 million times and store the modulo 2 of the randomly generated numbers in an array.
2. Store the first 127 output in a separate array and generate the rest of the array using A[i] = A[i-1] ^ A[i-127].
3. Sum the number of 1s and 0s for both the arrays which satisfy the condition and calculate the probability.

Sample Output:

For rand() modulo 2 the probability
P(Xi = 0/Xi-1 = 0) = 0.500110
P(Xi = 0/Xi-1 = 1) = 0.499770
For XOR Generator the probability
P(Xi = 0/Xi-1 = 0) = 0.499424
P(Xi = 0/Xi-1 = 1) = 0.500018

Conclusion:
For both arrays the probability is similar.

Question 5(C):

Part 1: encryption

1. Start by making for loop run for 127 times and store the modulo 2 of the randomly generated numbers in an array, output this into a file called key.txt.
2. Store the first 127 output in a separate array and generate the rest of the key using A[i] = A[i-1] ^ A[i-127].
3. Input the data which you want to encrypt using fopen(), type cast it into integer and convert it into binary.
4. Now print out Y = Data[i] ^ Key[i+127] into a file called crypt.txt

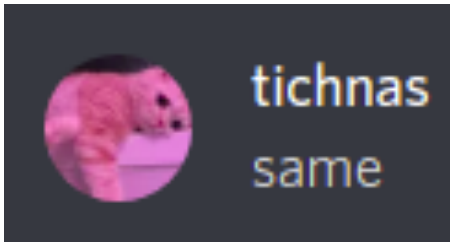
Part 2: decryption

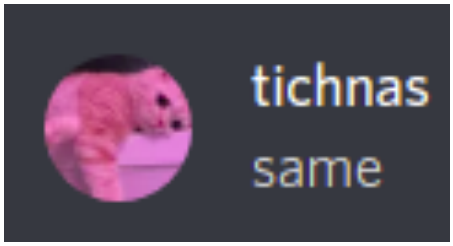
1. Start by inputting the key and crypted file.
2. Using the first 127 output in a separate array and generate the rest of the key using

A[i] = A[i-1] ^ A[i-127].

3. Now take the exor of the key with the data from the crypted file, then convert 8 bits at a time to integer values.
4. Type case the int values into char and output the values into decrypt.txt

Sample input & output:





Conclusion:

Any piece of information in any form can be easily coded and decoded for a safe transfer, since the possibilities for the key required ranges up to 2^127 so this makes it impossible to “guess” the key, therefore uncrackable.