Section 3 Report:

Cryptography and Bit Manipulation

Question 5

In our program, we make an XOR generator (XORG) in the following way:

- We use random number generation to pick a random 127-bit seed x_1, x_2, x_3, …. , x_{126}, x_{127}
- The subsequent bits, if required, are constructed using the following principle:

$$x_i = x_{i-1} \oplus x_{i-127} \text{ for } i > 127$$

Part A:

We wrote a program to generate one million bits with randomly assigned values of either 0 or 1. Then the probability distribution of 0s and 1s is calculated using the following formula:

$$P(E) = \frac{n(E)}{n(S)}$$

Here, n(S) = 1e6 while we have two cases for E:

- Event of getting 1 as a bit
- Event of getting 0 as a bit

We notice that P(E) for both cases turn out to be very close to 0.5 which is also what should be ideally happening.

Part B:

Then, we wrote a program to use the same approach but this time to compute two different things implementing the formula for conditional probability:

$$P(\frac{A}{B}) = \frac{P(A \cap B)}{P(B)}$$

We use this in our code to compute two cases:

- $P(\frac{x_i=0}{x_{i-1}=0})$
- $P(\frac{x_i=0}{x_{i-1}=1})$

We observe that both the conditional probabilities turn out to be very close to 0.5 which is in line with what should be happening theoretically.

Part C:

This part involved the encryption and decryption of files.

Encryption:

We wrote a program to firstly obtain the binary representation of the inputted file. Then the XOR generator (mentioned above) was used to encrypt the data bits (b_i) using the following principle:

$$e_i = b_i \oplus x_{i+127}$$

Decryption:

We wrote a program to firstly read the encrypted bits and then the XOR generator (mentioned above) is used to encrypt the data bits (b_i) using the following principle:

$$b_i = e_i \oplus x_{i+127}$$

Now, these binary bits are converted back to their normal characters and written to an output file with appropriate extension and then, we get back our decrypted original file.