NAME: SUBHAPREET PATRO

ROLL NO.: 2211CS10547

GROUP: 3

Week-1

Aim:

Program for the execution of XOR Encryption in Java.

Description:

The XOR Encryption algorithm is a very effective yet easy to implement method of symmetric encryption. Due to its effectiveness and simplicity, the XOR Encryption is an extremely common component used in more complex encryption algorithms used nowadays.

The XOR encryption algorithm is an example of symmetric encryption where the same key is used to both encrypt and decrypt a message.

The XOR Encryption algorithm is based on applying an XOR mask using the plaintext and a key.

Reapplying the same XOR mask (using the same key) to the cipher text outputs the original plain text. The following truth table (based on the XOR truth table) demonstrates how the encryption process works.

The XOR encryption algorithm can be applied to any digital/binary information, included text based information encoded using the 8-bit ASCII code. In this case the encryption key can be expressed as a string of characters.

By itself, the XOR encryption can be very robust if:

- It is based on a long key that will not repeat itself. (e.g. a key that contains as many bits/characters as the plaintext)
- A new key is randomly generated for any new communication.
- The key is kept secret by both the sender and the receiver.

When a large quantity of text is to be encrypted, a shorter repeating encryption key is used to match the length of the plain text. However re-using the same key over and over, or using a shorter repeating key results in a less secure method where the cipher text could be decrypted using a frequency analysis.

Example:

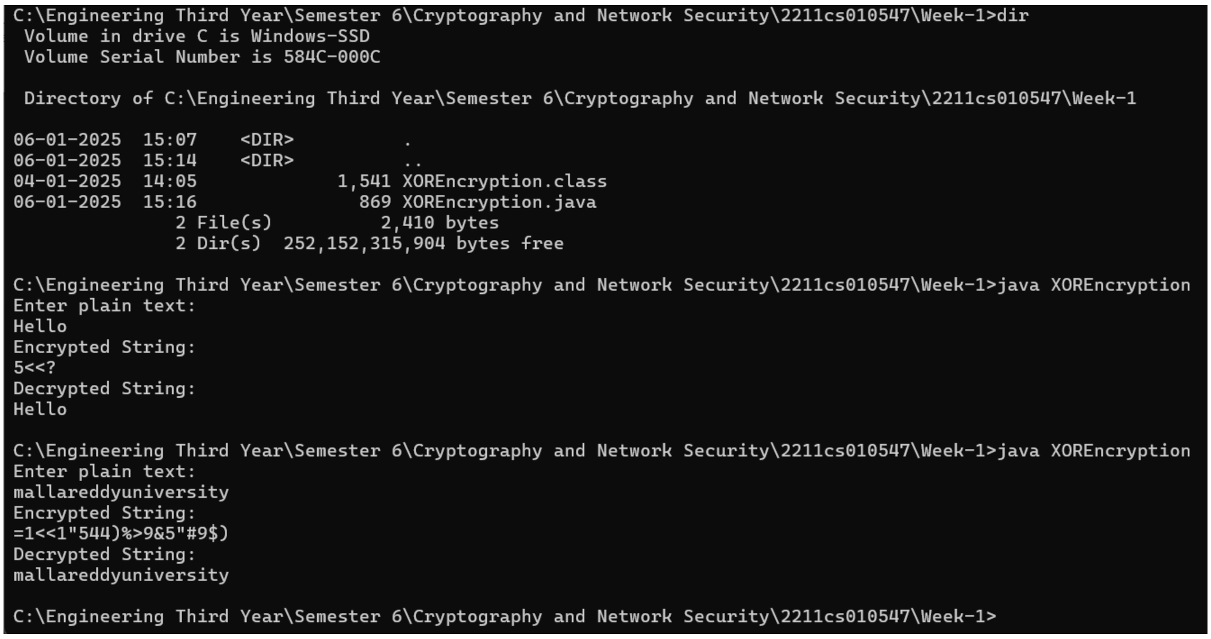
Input: A →01000001
Key: K →01001011
XOR ENCRYPTION
Output: →00001010

Program:

```
import java.util.Scanner;
public class XOREncryption {
    public static String encryptDecrypt(String inputString) {
        char xorKey = 'P';
        String outputString = "";
        int len = inputString.length();
        for (int i = 0; i < len; i++) {
            outputString = outputString + Character.toString((char)
(inputString.charAt(i) ^ xorKey));
        }
        System.out.println(outputString);
        return outputString;
    }

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter plain text: ");
        String sampleString = s.nextLine();
        System.out.println("Encrypted String: ");
        String encryptedString = encryptDecrypt(sampleString);
        System.out.println("Decrypted String: ");
        encryptDecrypt(encryptedString);
    }
}
```

Output:



```
C:\Engineering Third Year\Semester 6\Cryptography and Network Security\2211cs010547\Week-1>dir
 Volume in drive C is Windows-SSD
 Volume Serial Number is 584C-000C

 Directory of C:\Engineering Third Year\Semester 6\Cryptography and Network Security\2211cs010547\Week-1

06-01-2025  15:07    <DIR>          .
06-01-2025  15:14    <DIR>          ..
04-01-2025  14:05             1,541 XOREncryption.class
06-01-2025  15:16               869 XOREncryption.java
               2 File(s)          2,410 bytes
               2 Dir(s)  252,152,315,904 bytes free

C:\Engineering Third Year\Semester 6\Cryptography and Network Security\2211cs010547\Week-1>java XOREncryption
Enter plain text:
Hello
Encrypted String:
5<<?
Decrypted String:
Hello

C:\Engineering Third Year\Semester 6\Cryptography and Network Security\2211cs010547\Week-1>java XOREncryption
Enter plain text:
mallareddyuniversity
Encrypted String:
=1<<1"544)%>9&5"#9$)
Decrypted String:
mallareddyuniversity

C:\Engineering Third Year\Semester 6\Cryptography and Network Security\2211cs010547\Week-1>
```