Assignment 1
Implementation of Caesar Cipher/Additive Cipher

PRN No: 2018BTECS00212

Aim: Implementation of Caesar Cipher/Additive Cipher using ASCII Substitution as a key for encryption and decryption.

Theory:
The Caesar cipher is one of the oldest and simplest ciphers. It is used to eliminate the security issues like man in the middle or masquerade. It is a type of substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places down the alphabet. For example, with a shift of 2, A would be replaced by C, B would become D, and so on. The method is named after Julius Caesar, who apparently used it to communicate with his generals.
There are also some issues with this cipher regarding vulnerability of the key, so the key must be only known to the sender and receiver, else the other person listening to the conversation can simply decrypt the message. We must have an efficient key distribution system to do so. Kerberos key distribution is one of the techniques.
In the given assignment we are using ASCII substitution as a key.ASCII is an acronym for American Standard Code for Information Interchange.It is a code that uses numbers to represent characters. Each letter is assigned a number between 0 and 127. Upper and lower case characters are assigned different numbers. For example the character A is assigned the decimal number 65, while a is assigned decimal 97.
For example if plaintext is **WCESANGLI**, here W is converted into 87, C into 67 and so on. Encrypted text would be **876769836578717673** and while decrypting we use the same key approach.

Algorithm:

Encryption algorithm:

```
ceasers_encryption(plainText)
        Initialize cipherText to “”
        foreach character in plainText
        do
                set cipherText to cipherText + ASCII_CODE(character)
        done
return cipherText
end ceasers_encryption
```

Decryption algorithm:

ceasers_decryption(cipherText)
Initialize plainText to “”
Foreach character in cipherText
do
    If character is secondLast character of cipherText
        break
    end If
    Initialize charAscii to “”
    Initialize AsciiSum to ASCII_CODE(character)+ASCII_CODE(next_character)
    If AsciiSum greater than 65
        set charAscii to AsciiSum + next_character’s next character Ascii
        Jump by three characters
    Else
        Set charAscii to AsciiSum
        Jump by two characters
    End if
    append character whose ascii is charAscii to plainText
return plainText
end ceasers_decryption

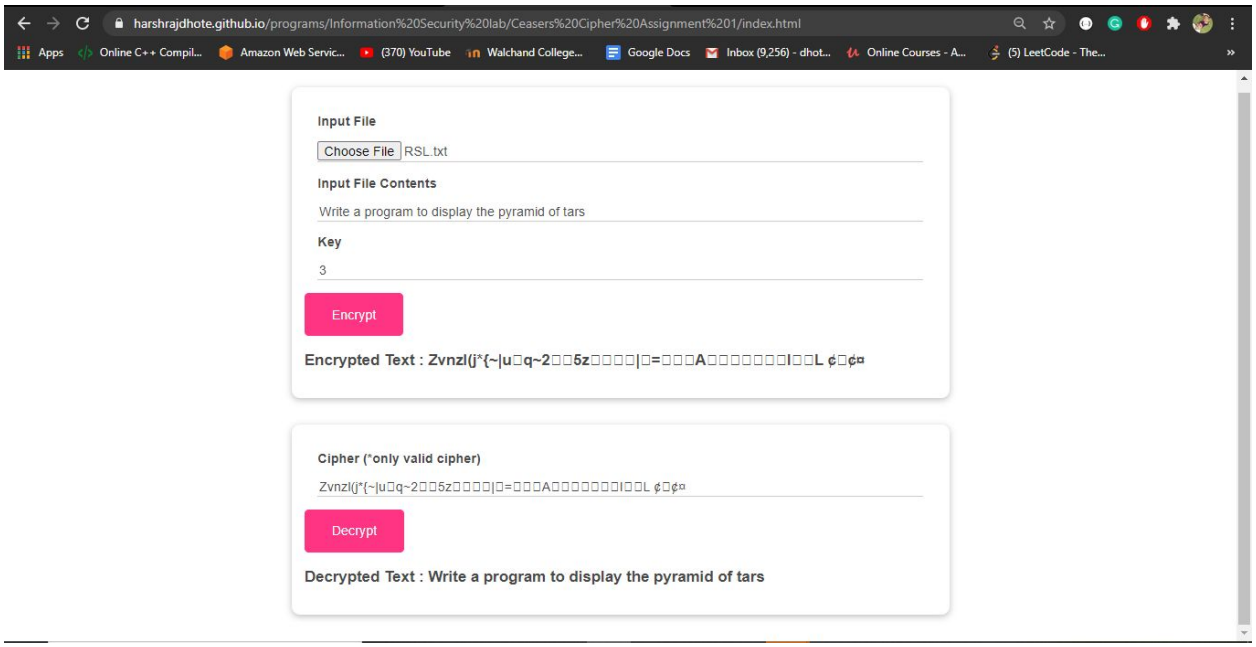
Source Code:

```
let plainText = document.getElementById('plainText');

let decryptBtn = document.getElementById('decrypt-btn');
let encryptBtn = document.getElementById('encrypt-btn');
let encryptResult = document.getElementById('encryptResult');
let decryptResult = document.getElementById('decryptResult');

document.getElementById('inputFile')
            .addEventListener('change', function() {

            var fr=new FileReader();
            fr.onload=function(){
                document.getElementById('plainText')
```

```
                        .value=fr.result;
                console.log(fr.result);
            }

            fr.readAsText(this.files[0]);
        })
class CeasersCipher{
    static encrypt() {
        let key = document.getElementById('key').value;
        let PlainText = plainText.value;
        let cipherText="";
        let PlainTextSize = PlainText.length;
        key = +key;
        for(let i=0;i<PlainTextSize;i++){
            cipherText += String.fromCharCode((PlainText.charCodeAt(i) +
(+key))%256);
            key++;
         }
        console.log(cipherText);

         encryptResult.innerHTML = `<h3>Encrypted Text :
${cipherText}</h3>`;
    }

    static decrypt(){
        let cipherText = document.getElementById('cipherText');
        let key = document.getElementById('key').value;
        let CipherText = cipherText.value;
        let plainText="";
        let plainTextSize = CipherText.length;
              let i = 0;
        while(i<plainTextSize){
            console.log(CipherText)
            plainText += String.fromCharCode((CipherText.charCodeAt(i) -
(+key))%256);
            key++;
            i++;
```

```
        }
        console.log(plainText);
        decryptResult.innerHTML = `<h3>Decrypted Text :
${plainText}</h3>`;

    }
}
encryptBtn.addEventListener('click',CeasersCipher.encrypt);
decryptBtn.addEventListener('click',CeasersCipher.decrypt);
```

Demo Link



Conclusion:

Hence I successfully implemented the Caesars Cipher/Additive Cipher using ASCII Substitution as a key for encryption and decryption.