Practical Assignment - 2

AIM: Implementation of RSA algorithm in JavaScript.

Code:

```
function gcd(a, b) {
    while (b !== 0) {
        let temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

function modInverse(a, m) {
    let m0 = m, t, q;
    let x0 = 0, x1 = 1;

    if (m === 1) return 0;

    while (a > 1) {
        q = Math.floor(a / m);
        t = m;

        m = a % m;
        a = t;
        t = x0;

        x0 = x1 - q * x0;
        x1 = t;
    }

    if (x1 < 0) x1 += m0;

    return x1;
}

// Modular exponentiation function
function modExp(base, exp, mod) {
    let result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 === 1) {
            result = (result * base) % mod;
        }
        exp = Math.floor(exp / 2);
```

```
        base = (base * base) % mod;
    }
    return result;
}

// Key Generation
function generateKeys() {
    const p = 61; // First prime
    const q = 53; // Second prime

    const n = p * q;
    const phi = (p - 1) * (q - 1);

    // Public exponent
    let e = 17; // Common choice

    // Ensure e and phi are coprime
    while (gcd(e, phi) !== 1) {
        e++;
    }

    // Private exponent
    const d = modInverse(e, phi);

    return {
        publicKey: { e, n },
        privateKey: { d, n }
    };
}

// Encryption
function encrypt(publicKey, plaintext) {
    const { e, n } = publicKey;
    let encrypted = '';

    for (let i = 0; i < plaintext.length; i++) {
        const charCode = plaintext.charCodeAt(i);
        const encryptedChar = modExp(charCode, e, n);
        encrypted += String.fromCharCode(encryptedChar);
    }

    return encrypted;
}

// Decryption
function decrypt(privateKey, ciphertext) {
    const { d, n } = privateKey;
    let decrypted = '';

    for (let i = 0; i < ciphertext.length; i++) {
        const charCode = ciphertext.charCodeAt(i);
        const decryptedChar = modExp(charCode, d, n);
```

```
        decrypted += String.fromCharCode(decryptedChar);
    }

    return decrypted;
}

// Example Usage
const { publicKey, privateKey } = generateKeys();

console.log('Public Key:', publicKey);
console.log('Private Key:', privateKey);

const message = 'Shaurya123#*&%`~';
console.log('Original Message:', message);

const encryptedMessage = encrypt(publicKey, message);
console.log('Encrypted Message:', encryptedMessage);

const decryptedMessage = decrypt(privateKey, encryptedMessage);
console.log('Decrypted Message:', decryptedMessage);
```

Output: