**LAB 04**

**ENCRYPT:**

function playfair(plain, key) {

}

playfair.prototype.encrypt = function(plain, key) {

key = constructKeyMatrix(key);

plain = editPlain(plain);

result = encryptPlayFair(plain, key);

return result;

}

function constructKeyMatrix(key) {

const alphabet = "abcdefghiklmnopqrstuvwxyz";

key += alphabet;

for (let i = 0; i < key.length; i++) {

if (key.indexOf(key[i]) !== i) {

key = key.slice(0, i) + key.slice(i + 1);

i--;

}

}

var k = 0;

var matrix = new Array(5);

for (var i = 0; i < 5; i++) {

matrix[i] = new Array(5);

}

for (var i = 0; i < 5; i++) {

for (var j = 0; j < 5; j++) {

matrix[i][j] = key[k];

k++;

}

}

console.log(matrix);

return key;

}

function editPlain(plain) {

for (let i = 0; i < plain.length - 1; i += 2) {

if (plain[i] === plain[i + 1])

plain = plain.slice(0, i + 1) + 'x' + plain.slice(i + 1);

}

if (plain.length % 2 === 1) plain += 'x';

plain = plain.replace(/j/g, 'i');

return plain;

}

function encryptPlayFair(plaintext, key) {

var ciphertext = "";

for (let i = 0; i < plaintext.length - 1; i += 2) {

var i1, i2, j1, j2;

i1 = key.indexOf(plaintext[i]) / 5 | 0;

j1 = key.indexOf(plaintext[i]) % 5;

i2 = key.indexOf(plaintext[i + 1]) / 5 | 0;

j2 = key.indexOf(plaintext[i + 1]) % 5;

if (i1 == i2)

ciphertext += key[i1 \* 5 + (j1 + 1) % 5] + key[i2 \* 5 + (j2 + 1) % 5];

else if (j1 == j2)

ciphertext += key[((i1 + 1) % 5) \* 5 + j1] + key[((i2 + 1) % 5) \* 5 + j2];

else

ciphertext += key[i1 \* 5 + j2] + key[i2 \* 5 + j1];

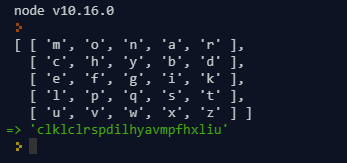
}

return ciphertext;

}

playfair.prototype.encrypt("meetmeattheschoolhouse","monarchy")

**OUTPUT:**



**DECRYPT:**

function playfair(plain, key) {

}

playfair.prototype.encrypt = function(plain, key) {

key = constructKeyMatrix(key);

plain = editPlain(plain);

result = encryptPlayFair(plain, key);

return result;

}

function constructKeyMatrix(key) {

const alphabet = "abcdefghiklmnopqrstuvwxyz";

key += alphabet;

for (let i = 0; i < key.length; i++) {

if (key.indexOf(key[i]) !== i) {

key = key.slice(0, i) + key.slice(i + 1);

i--;

}

}

var k = 0;

var matrix = new Array(5);

for (var i = 0; i < 5; i++) {

matrix[i] = new Array(5);

}

for (var i = 0; i < 5; i++) {

for (var j = 0; j < 5; j++) {

matrix[i][j] = key[k];

k++;

}

}

console.log(matrix);

return key;

}

function editPlain(plain) {

for (let i = 0; i < plain.length - 1; i += 2) {

if (plain[i] === plain[i + 1])

plain = plain.slice(0, i + 1) + 'x' + plain.slice(i + 1);

}

if (plain.length % 2 === 1) plain += 'x';

plain = plain.replace(/j/g, 'i');

return plain;

}

function encryptPlayFair(plaintext, key) {

var ciphertext = "";

for (let i = 0; i < plaintext.length - 1; i += 2) {

var i1, i2, j1, j2;

i1 = key.indexOf(plaintext[i]) / 5 | 0;

j1 = key.indexOf(plaintext[i]) % 5;

i2 = key.indexOf(plaintext[i + 1]) / 5 | 0;

j2 = key.indexOf(plaintext[i + 1]) % 5;

if (i1 == i2)

ciphertext += key[i1 \* 5 + (j1 - 1) % 5] + key[i2 \* 5 + (j2 - 1) % 5];

else if (j1 == j2)

ciphertext += key[((i1 - 1) % 5) \* 5 + j1] + key[((i2 - 1) % 5) \* 5 + j2];

else

ciphertext += key[i1 \* 5 + j2] + key[i2 \* 5 + j1];

}

return ciphertext;

}

playfair.prototype.encrypt("clklclrspdilhyavmpfhxliu","monarchy")

**OUTPUT:**

