function decodeYValue(base, value) {

// Convert the value to an integer from the specified base

return parseInt(value, base);

}

function findCoefficients(points) {

// Create a matrix to represent the equations

const matrix = [];

const n = points.length;

for (let i = 0; i < n; i++) {

const row = [];

const x = points[i][0];

const y = points[i][1];

for (let j = n - 1; j >= 0; j--) {

row.push(Math.pow(x, j));

}

row.push(y);

matrix.push(row);

}

// Solve the system of equations using Gaussian elimination

for (let i = 0; i < n - 1; i++) {

for (let j = i + 1; j < n; j++) {

const factor = matrix[j][i] / matrix[i][i];

for (let k = i; k < n + 1; k++) {

matrix[j][k] -= factor \* matrix[i][k];

}

}

}

// Extract the coefficients from the last column

const coefficients = [];

for (let i = n - 1; i >= 0; i--) {

coefficients.push(matrix[i][n] / matrix[i][i]);

}

return coefficients;

}

function findSecret(points) {

// Decode the Y values

const decodedPoints = points.map(([x, yBase, yValue]) => [x, decodeYValue(yBase, yValue)]);

// Find the coefficients of the polynomial

const coefficients = findCoefficients(decodedPoints);

// The secret is the constant term (coefficient at index 0)

return coefficients[0];

}

// Example usage:

const testCases = [

{

"keys": {

"n": 4,

"k": 3

},

"1": {

"base": "10",

"value": "4"

},

"2": {

"base": "2",

"value": "111"

},

"3": {

"base": "10",

"value": "12"

},

"6": {

"base": "4",

"value": "213"

}

},

{

"keys": {

"n": 9,

"k": 6

},

"1": {

"base": "10",

"value": "28735619723837"

},

"2": {

"base": "16",

"value": "1A228867F0CA"

},

"3": {

"base": "12",

"value": "32811A4AA0B7B"

},

"4": {

"base": "11",

"value": "917978721331A"

},

"5": {

"base": "16",

"value": "1A22886782E1"

},

"6": {

"base": "10",

"value": "28735619654702"

},

"7": {

"base": "14",

"value": "71AB5070CC4B"

},

"8": {

"base": "9",

"value": "122662581541670"

},

"9": {

"base": "8",

"value": "642121030037605"

}

}

// Add more test cases here

];

for (const testCase of testCases) {

const points = Object.entries(testCase).filter(entry => entry[0] !== "keys").map(([key, value]) => [parseInt(key), value.base, value.value]);

const secret = findSecret(points);

console.log("Secret:", secret);

}