#include <iostream>

#include <fstream>

#include <string>

#include <map>

#include <vector>

#include <nlohmann/json.hpp> // nlohmann JSON library

using namespace std;

using json = nlohmann::json;

// Helper function to decode a string from any given base to decimal

long long decodeBase(const string& value, int base) {

long long result = 0;

for (char ch : value) {

result = result \* base + (isdigit(ch) ? ch - '0' : ch - 'a' + 10);

}

return result;

}

// Lagrange Interpolation to calculate the constant term (c)

long long lagrangeInterpolation(const vector<pair<int, long long>>& points) {

long long constantTerm = 0;

int k = points.size();

for (int i = 0; i < k; i++) {

long long xi = points[i].first;

long long yi = points[i].second;

// Compute the basis polynomial Li(0) at x = 0

long long term = yi;

for (int j = 0; j < k; j++) {

if (j != i) {

term \*= -points[j].first;

term /= (xi - points[j].first);

}

}

constantTerm += term;

}

return constantTerm;

}

int main() {

// Open and parse the JSON file

ifstream inFile("testcase.json");

if (!inFile) {

cerr << "Error: Could not open JSON file." << endl;

return 1;

}

json inputJson;

inFile >> inputJson;

vector<pair<int, long long>> points;

int n = inputJson["keys"]["n"];

int k = inputJson["keys"]["k"];

// Extract and decode each root

for (int i = 1; i <= n; i++) {

string key = to\_string(i);

if (inputJson.contains(key)) {

int x = stoi(key);

int base = inputJson[key]["base"];

string value = inputJson[key]["value"];

long long y = decodeBase(value, base);

points.push\_back({x, y});

}

}

// Apply Lagrange interpolation to find the constant term (c)

long long secretC = lagrangeInterpolation(points);

cout << "Secret (c): " << secretC << endl;

return 0;

}