#include <iostream>

#include <iomanip>

using namespace std;

#define MAX 20

int n; // Number of keys

float p[MAX], q[MAX]; // Probabilities

float wt[MAX][MAX], c[MAX][MAX]; // Weight and cost matrices

int r[MAX][MAX]; // Root matrix

void con\_obst() {

int i, j, k, l, root;

float minCost, cost;

// Initialization: diagonal values

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

wt[i][i] = q[i];

c[i][i] = 0;

r[i][i] = 0;

}

// l = length of subtree

for (l = 1; l <= n; l++) {

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

j = i + l;

wt[i][j] = wt[i][j - 1] + p[j] + q[j];

minCost = 999999; // Large value

root = -1;

for (k = i + 1; k <= j; k++) {

cost = c[i][k - 1] + c[k][j] + wt[i][j];

if (cost < minCost) {

minCost = cost;

root = k;

}

}

c[i][j] = minCost;

r[i][j] = root;

}

}

}

// Recursively print structure of OBST

void print(int l1, int r1, int parent, bool isLeft) {

if (l1 >= r1) return;

int root = r[l1][r1];

if (parent == -1)

cout << "Root of OBST: " << root << "\n";

else

cout << (isLeft ? "Left" : "Right") << " child of " << parent << " is " << root << "\n";

print(l1, root - 1, root, true); // Left subtree

print(root, r1, root, false); // Right subtree

}

int main() {

int i;

cout << "Enter number of keys: ";

cin >> n;

cout << "Enter probabilities for successful searches (p1 to pn): ";

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

cin >> p[i];

}

cout << "Enter probabilities for unsuccessful searches (q0 to qn): ";

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

cin >> q[i];

}

con\_obst();

cout << "\nWeight Matrix:\n";

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

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

cout << setw(6) << wt[i][j] << " ";

}

cout << "\n";

}

cout << "\nCost Matrix:\n";

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

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

cout << setw(6) << c[i][j] << " ";

}

cout << "\n";

}

cout << "\nRoot Matrix:\n";

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

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

cout << setw(6) << r[i][j] << " ";

}

cout << "\n";

}

cout << "\nOptimal Binary Search Tree Structure:\n";

print(0, n, -1, false);

return 0;

}