```
#include<iostream>
using namespace std;
void calculateOptimalBST(int n);
void printTree(int l1, int r1);
int n;
float a[50], b[50];
float wt[50][50], c[50][50];
int r[50][50];
void calculateOptimalBST(int n) {
    float min;
    for (int i = 0; i < n; i++) {
        c[i][i] = 0.0;
        r[i][i] = 0;
        wt[i][i] = b[i];
        wt[i][i + 1] = b[i] + b[i + 1] + a[i + 1];
        c[i][i + 1] = b[i] + b[i + 1] + a[i + 1];
        r[i][i + 1] = i + 1;
    }
    c[n][n] = 0.0;
    r[n][n] = 0;
    wt[n][n] = b[n];
    for(int i = 2; i <= n; i++) {
        for(int j = 0; j <= n - i; j++) {
            wt[j][j + i] = b[j + i] + a[j + i] + wt[j][j + i - 1];
            c[j][j + i] = 9999;
            for (int l = j + 1; l <= j + i; l++) {
                if (c[j][j + i] > (c[j][l - 1] + c[l][j + i])) {
                    c[j][j + i] = c[j][l - 1] + c[l][j + i];
                    r[j][j + i] = l;
                }
            }
            c[j][j + i] += wt[j][j + i];
        }
    }
    cout << endl;
    cout << "\nOptimal BST : ";
    cout << "\nw[0][" << n << "] : " << wt[0][n];
    cout << "\nc[0][" << n << "] : " << c[0][n];
    cout << "\nr[0][" << n << "] : " << r[0][n];
    cout << endl;
}

void printTree(int l1, int r1) {
    if (l1 >= r1)
        return;
    if (r[l1][r[l1][r1] - 1] != 0)
        cout << "\nLeft child of " << r[l1][r1] << " node is : " << r[l1][r[l1][r1] - 1];
    if (r[r[l1][r1]][r1] != 0)
        cout << "\nRight child of " << r[l1][r1] << " node is : " << r[r[l1][r1]][r1];
    printTree(l1, r[l1][r1] - 1);
    printTree(r[l1][r1], r1);
    return;
}
int main() {
    cout << "\nOptimal Binary Search Tree\n";
    cout << "\nEnter the number of nodes : ";
    cin >> n;
    cout << "\nProbability for Successful search ::\n\n";
    for (int i = 1; i <= n; i++) {
        cout << "p[" << i << "] : ";
        cin >> a[i];
    }
    cout << "\nProbability for Unsuccessful search ::\n\n";
    for (int i = 0; i <= n; i++) {
        cout << "q[" << i << "] : ";
        cin >> b[i];
    }
    calculateOptimalBST(n);
    printTree(0, n);
    cout << endl;
    return 0;
}
```