```
#include <iostream>
#include <climits>
using namespace std;
class OBST {
   int n;
   string a[10];
   int p[10], q[10];
   int w[10][10], c[10][10], r[10][10];
public:
   void get_data();
   void build_OBST();
   void build_tree(int, int);
};
void OBST::get_data() {
   cout << "Enter the number of nodes: ";
   cin >> n;
   cout << "Enter the data as...\n";
   for (int i = 1; i <= n; i++) {
      cout << "a[" << i << "]: ";
      cin >> a[i];
   }
   for (int i = 1; i <= n; i++) {
      cout << "p[" << i << "]: ";
      cin >> p[i];
   }
   for (int i = 0; i <= n; i++) {
      cout << "q[" << i << "]: ";
      cin >> q[i];
   }
}
void OBST::build_OBST()
```

```
    int i, j, k, l, m, min;
    for (i = 0; i < n + 1; i++) {
        w[i][i] = q[i];
        c[i][i] = 0;
        r[i][i] = 0;
        w[i][i + 1] = q[i] + p[i + 1] + q[i + 1];
        c[i][i + 1] = w[i][i + 1];
        r[i][i + 1] = i + 1;
    }
    for (m = 2; m <= n; m++) {
        for (i = 0; i <= n - m; i++) {
            j = i + m;
            w[i][j] = w[i][j - 1] + p[j] + q[j];
            min = INT_MAX;
            for (k = i + 1; k <= j; k++) {
                if ((c[i][k - 1] + c[k][j]) < min) {
                    min = c[i][k - 1] + c[k][j];
                    r[i][j] = k;
                }
            }
            c[i][j] = min + w[i][j];
        }
    }
    cout << "The Optimal Binary Search Tree For the Given Nodes Is...\n";
    cout << "The Root of this OBST is: " << r[0][n] << endl;
    cout << "The Cost of this OBST is: " << c[0][n] << endl;
    cout << "NODE\tLEFT CHILD\tRIGHT CHILD\n";
    build_tree(0, n);
}
void OBST::build_tree(int i, int j) {
    int root = r[i][j];
```

```
if (root == 0) return;
   cout << a[root] << "\t";
   if (r[i][root - 1] != 0) cout << a[r[i][root - 1]] << "\t\t";
   else cout << "-\t\t";
   if (r[root][j] != 0) cout << a[r[root][j]] << "\n";
   else cout << "-\n";
   build_tree(i, root - 1);
   build_tree(root, j);
}
int main() {
   OBST obj;
   obj.get_data();
   obj.build_OBST();
   return 0;
}
```

Output:

```
Enter the number of nodes: 3
Enter the data as...
a[1]: 10        a[2]: 20        a[3]: 30
p[1]: 4         p[2]: 2         p[3]: 6
q[0]: 3
q[1]: 3
q[2]: 1
q[3]: 1
The Optimal Binary Search Tree For the Given Nodes Is...
The Root of this OBST is: 3
The Cost of this OBST is: 36
NODE    LEFT CHILD      RIGHT CHILD
30      10      -
10      -       20
20      -       -
```