#include<iostream>

#include<limits.h>

using namespace std;

struct Node{

int key;

Node\* left, \*right;

Node(int k) {

key = k;

left = NULL;

right = NULL;

}

};

//function to find optimal root

int findOptimalRoot(int keys[], int freq[], int start, int end) {

int sum = 0;

for(int i = start; i<=end; i++) {

sum += freq[i];

}

int minCost = INT\_MAX;

int cost;

for(int i = start; i<=end; i++) {

cost = sum+((i>start) ? findOptimalRoot(keys, freq, start, i-1):0)

+ ((i<end) ? findOptimalRoot(keys, freq, i+1, end):0);

if(cost < minCost) {

minCost = cost;

}

}

return minCost;

}

//function to construct and optimal bst

Node\* buildOptimalBST(int keys[], int freq[], int start, int end) {

if(start > end) {

return NULL;

}

int minIdx;

int minCost = INT\_MAX;

int cost;

for(int i = start; i<= end; i++) {

cost = findOptimalRoot(keys, freq, start, end);

if(cost < minCost) {

minCost = cost;

minIdx = i;

}

}

Node\* root = new Node(keys[minIdx]);

root->left = buildOptimalBST(keys, freq, start, minIdx-1);

root->right = buildOptimalBST(keys, freq, minIdx+1, end);

return root;

}

void inorder(Node\* root) {

if(root == NULL) {

return;

}

inorder(root->left);

cout<<root->key<<" ";

inorder(root->right);

}

int main() {

int k;

cout<<"\n ENter number of keys: ";

cin>>k;

cout<<"\n Enter keys: ";

int keys[k];

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

cin>>keys[i];

}

cout<<"\n Enter the frequencies of keys: ";

int freq[k];

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

cin>>freq[i];

}

int n = sizeof(keys)/sizeof(keys[0]);

Node\* root = buildOptimalBST(keys, freq, 0, n-1);

cout<<"\n Inorder traversal of the optimal binary search tree: ";

inorder(root);

return 0;

}