1

#include<iostream>

using namespace std;

int fib(int n)

{

int a, b;

if (n < 2)

return 1;

else

{

a = 1, b = 1;

for (int i = 0; i < n - 1; i++)

{

a = a + b;

b = a - b;

}

}

return a;

}

int main()

{

int n;

cin >> n;

cout << fib(n);

return 0;

}

2

#include<iostream>

using namespace std;

int m[20][20];

int s[20][20];

void maxchain(int p[], int m[][20], int s[][20], int n)

{

for (int i = 1; i <= n; i++)m[i][i] = 0;

for (int r = 2; r <= n; r++)

for (int i = 1; i <= n - r + 1; i++)

{

int j = i + r - 1;

m[i][j] = m[i + 1][j] + p[i - 1] \* p[i] \* p[j];

s[i][j] = i;

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

{

int t = m[i][k] + m[k + 1][j] + p[i - 1] \* p[k] \* p[j];

if (t < m[i][j])

{

m[i][j] = t;

s[i][j] = k;

}

}

}

}

void t(int s[][20], int i, int j)

{

if (i == j)

cout << "A" << i;

else if (i + 1 == j)

cout << "(A" << i << "A" << j << ")";

else

{

cout << "(";

t(s, i, s[i][j]);

t(s, s[i][j] + 1, j);

cout << ")";

}

}

int main()

{

int n;

cin >> n;

int \*p = new int[n + 1];

for (int i = 0; i < n + 1; i++)

cin >> p[i];

maxchain(p, m, s, n);

t(s, 1, n);

}

3#include<iostream>

using namespace std;

int m[20][20];

int min(int a, int b)

{

if (a < b)

return a;

else

return b;

}

int max(int a, int b)

{

if (a < b)

return b;

else

return a;

}

void bb(int n, int c, int w[], int v[]) {

int jMax = min(w[n] - 1, c);

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

m[n][j] = 0;

}

for (int j = w[n]; j <= c; j++) {

m[n][j] = v[n];

}

for (int i = n - 1; i> 1; i--) {

jMax = min(w[i] - 1, c);

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

m[i][j] = m[i + 1][j];

}

for (int j = w[i]; j <= c; j++) {

m[i][j] = max(m[i + 1][j], m[i + 1][j - w[i]] + v[i]);

}

}

m[1][c] = m[2][c];

if (c >= w[1])

m[1][c] = max(m[1][c], m[2][c - w[1]] + v[1]);

}

void q(int n, int c, int w[], int x[])

{

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

if (m[i][c] == m[i + 1][c])

x[i] = 0;

else {

x[i] = 1;

c -= w[i];

}

x[n] = (m[n][c] > 0) ? 1 : 0;

}

}

int main()

{

int n;

cin >> n; //个数

int c;

cin >> c; //容量

int\*w = new int[n + 1];

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

{

cin >> w[i];

}

int \*v = new int[n + 1];

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

{

cin >> v[i];

}

bb(n, c, w, v);

int\* x = new int[n + 1];

q(n, c, w, x);

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

cout << x[i] << " ";

}

}

#include<iostream>

using namespace std;

int s[20][20];

int w[20][20];

int m[20][20];

void OptimalBinarySerachTree(int n, int b[], int a[])

{

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

{

w[i + 1][i] = a[i];

m[i + 1][i] = 0;

}

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

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

int j = i + r;

w[i][j] = w[i][j - 1] + a[j] + b[j];

m[i][j] = m[i + 1][j];

s[i][j] = i;

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

int t = m[i][k - 1] + m[k + 1][j];

if (t<m[i][j]){

m[i][j] = t;

s[i][j] = k;

}

}

m[i][j] += w[i][j];

}

}

}

void Traceback(int i, int j)

{

if (j>i){

int root = s[i][j];

cout << "S" << root << "是根" << endl;

if (s[i][root - 1]>0)

{

cout << "S" << root << "的左孩子是S" << s[i][root - 1] << endl;

}

if (s[root + 1][j]>0)

{

cout << "S" << root << "的右孩子是S" << s[root + 1][j] << endl;

}

Traceback(i, root - 1);

Traceback(root + 1, j);

}

}

int main()

{

int n;

cin >> n;

int \*b = new int[n + 1];

int \*a = new int[n + 1];

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

{

cin >> b[i];

}

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

{

cin >> a[i];

}

OptimalBinarySerachTree(n, b, a);

Traceback(1, n);

}