/\* This program is to implement optimal binary search tree\*/

#include<iostream>

using namespace std;

#define SIZE 10

class OBST

{

int p[SIZE]; // Probabilities with which we search for an element

int q[SIZE];//Probabilities that an element is not found

int a[SIZE];//Elements from which OBST is to be built

int w[SIZE][SIZE];//Weight ‘w[i][j]’ of a tree having root

//’r[i][j]’

int c[SIZE][SIZE];//Cost ‘c[i][j] of a tree having root ‘r[i][j]

int r[SIZE][SIZE];//represents root

int n; // number of nodes

public:

/\* This function accepts the input data \*/

void get\_data()

{

int i;

cout<<"\n Optimal Binary Search Tree \n";

cout<<"\n Enter the number of nodes";

cin>>n;

cout<<"\n Enter the data as…\n";

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

{

cout<<"a["<<i<<"]";

cin>>a[i];

}

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

{

cout<<"\n p["<<i<<"]";

cin>>p[i];

}

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

{

cout<<"\n q["<<i<<"]";

cin>>q[i];

}

}

/\* This function returns a value in the range ‘r[i][j-1]’ to ‘r[i+1][j]’so

that the cost ‘c[i][k-1]+c[k][j]’is minimum \*/

int Min\_Value(int i,int j)

{

int m,k;

int minimum=32000;

for(m=r[i][j-1];m<=r[i+1][j];m++)

{

if((c[i][m-1]+c[m][j])<minimum)

{

minimum=c[i][m-1]+c[m][j];

k=m;

}

}

return k;

}

/\* This function builds the table from all the given probabilities It

basically computes C,r,W values \*/

void build\_OBST()

{

int i,j,k,l,m;

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

{

//initialize

w[i][i]=q[i];

r[i][i]=c[i][i]=0;

//Optimal trees with one node

w[i][i+1]=q[i]+q[i+1]+p[i+1];

r[i][i+1]=i+1;

c[i][i+1]=q[i]+q[i+1]+p[i+1];

}

w[n][n]=q[n];

r[n][n]=c[n][n]=0;

//Find optimal trees with ‘m’ nodes

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];

k=Min\_Value(i,j);

c[i][j]=w[i][j]+c[i][k-1]+c[k][j];

r[i][j]=k;

}

}

}

/\* This function builds the tree from the tables made by the OBST function \*/

void build\_tree()

{

int i,j,k;

int queue[20],front=-1,rear=-1;

cout<<"The Optimal Binary Search Tree For the Given Node Is…\n";

cout<<"\n The Root of this OBST is ::"<<r[0][n];

cout<<"\nThe Cost of this OBST is::"<<c[0][n];

cout<<"\n\n\t NODE \t LEFT CHILD \t RIGHT CHILD ";

cout<<"\n";

queue[++rear]=0;

queue[++rear]=n;

while(front!=rear)

{

i=queue[++front];

j=queue[++front];

k=r[i][j];

cout<<"\n\t"<<k;

if(r[i][k-1]!=0)

{

cout<<"\t\t"<<r[i][k-1];

queue[++rear]=i;

queue[++rear]=k-1;

}

else

cout<<"\t\t";

if(r[k][j]!=0)

{

cout<<"\t"<<r[k][j];

queue[++rear]=k;

queue[++rear]=j;

}

else

cout<<"\t";

}//end of while

cout<<"\n";

}

};//end of the class

/\*This is the main function \*/

int main()

{

OBST obj;

obj.get\_data();

obj.build\_OBST();

obj.build\_tree();

return 0;

}