```
#include<iostream>
using namespace std;
#define SIZE 10
class OBST
{
int probsearch[SIZE];
int probabcnt[SIZE];
int arr[SIZE];
int weight[SIZE][SIZE];
int cost[SIZE][SIZE];
int root[SIZE][SIZE];
int n; // number of nodes
public:

void accept()
{
int i;
cout<<"\n Optimal Binary Search Tree \n";
cout<<"\n Enter the number of nodes :";
cin>>n;
cout<<"\n Enter the data : \n";
for(i=1;i<=n;i++)
{
cout<<"\n arr["<<i<<"] ";
cin>>arr[i];
}
for(i=1;i<=n;i++)
{
cout<<"\n probsearch["<<i<<"] ";
cin>>probsearch[i];
}

for(i=0;i<n;i++)
{
cout<<"\nprobabcnt["<<i<<"] ";
cin>>probabcnt[i];
}
}

int Get_min(int i,int j)
{
   int m,k;
int minval=32000;
for(m=root[i][j-1];m<=root[i+1][j];m++)
{
if((cost[i][m-1]+cost[m][j])<minval)
{
minval=cost[i][m-1]+cost[m][j];
k=m;
}
}
return k;
}

void make_OBST()
```

```
{
int i,j,k,l,m;
for(i=0;i<n;i++)
{
weight[i][i]=probabcnt[i];
root[i][i]=cost[i][i]=0;
weight[i][i+1]=probabcnt[i]+probabcnt[i+1]+probsearch[i+1];
root[i][i+1]=i+1;
cost[i][i+1]=probabcnt[i]+ probabcnt[i+1]+probsearch[i+1];
}
weight[n][n]=probabcnt[n];
root[n][n]=cost[n][n]=0;
for(m=2;m<=n;m++)
{
for(i=0;i<=n-m;i++)
{
j=i+m;
weight[i][j]=weight[i][j-1]+probsearch[j]+probabcnt[j];
k=Get_min(i,j);
cost[i][j]=weight[i][j]+cost[i][k-1]+cost[k][j];
root[i][j]=k;
}
}
}

void build_tree()
{

int i,j,k;
int q[20],front=-1,rear=-1;
cout<<"\nThe Optimal Binary Search Tree For the Given Node Is...\n";
cout<<"\n The Root of this OBST is "<<root[0][n];
cout<<"\nThe Cost of this OBST is:"<<cost[0][n];
cout<<"\n\n\t NODE \t LEFT CHILD \tRIGHT CHILD";
cout<<"\n";
q[++rear]=0;
q[++rear]=n;
while(front!=rear)
{
i=q[++front];
j=q[++front];
k=root[i][j];
cout<<"\n\t"<<k;
if(root[i][k-1]!=0)
{
cout<<"\t\t"<<root[i][k-1];
q[++rear]=i;
q[++rear]=k-1;
}
else
cout<<"\t\t";
if(root[k][j]!=0)
{
cout<<"\t"<<root[k][j];
q[++rear]=k;
```

```
q[++rear]=j;
}
else
cout<"\t";
}
cout<<"\n";
}
};

int main()
{

OBST tst;
tst.accept();
tst.make_OBST();
tst.build_tree();
return 0;
}
```