```
//============================================================================
// Name        : OBST.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
#include<string>
using namespace std;

class BTNODE
{
	BTNODE *left;
	string word;
	BTNODE *right;
public:
	BTNODE()
    {
	  left=right=NULL;
	  word=" ";
    }
	friend class OBST;
```

```
};

class OBST
{
        BTNODE * Root;
   int n;
   float p[10],q[10];
   char words[10][20];
public:
        OBST()
   {
          Root=NULL;
          n=0;
   }
        void accept_data();
        void Optimal_BST();
   int find_min(float c[10][10],int,int);
   BTNODE * construct(int r[10][10],int,int);
   void preorder(BTNODE* T);
   friend int main();
};

void OBST::accept_data()
{
   cout<<"\n Enter the no of word=";
```

```
    cin>>n;
    cout<<"\n Enter the words in sorted order=>";
    for(int i=1;i<=n;i++)
    {
        cin>>words[i];
    }

    cout<<"\n Enter the successful search probabilities (P)=";
    for(int i=1;i<=n;i++)
    {
        cin>>p[i];
    }

    cout<<"\n Enter the unsuccessful search probabilities (q)=";
    for(int i=0;i<=n;i++)
    {
        cin>>q[i];
    }
}

void OBST::Optimal_BST()
{
        float c[10][10],w[10][10];
        int r[10][10],i,j,k,slot;
```

```
     for(i=0;i<10;i++)
     {
          for(j=0;j<10;j++)
          {
               c[i][j]=w[i][j]=r[i][j]=0;
          }
     }

     for(i=1;i<=n;i++)
     {
          w[i][i]=q[i-1]+q[i]+p[i];
          c[i][i]=w[i][i];
          r[i][i]=i;
     }

for(slot=2;slot<=n;slot++)
{
     for(i=1;i<=n-slot+1;i++)
     {
          j=i+slot-1;
          w[i][j]=w[i][j-1]+p[j]+q[j];
          k=find_min(c,i,j);
          c[i][j]=w[i][j]+c[i][k-1]+c[k+1][j];
          r[i][j]=k;
     }
```

```
    }

    Root=construct(r,1 ,n);
}

int OBST::find_min(float c[10][10],int i,int j)
{
    float min=999.99;
    int l,k;
    for(k=i;k<=j;k++)
    {
          if((c[i][k-1]+c[k+1][j])<min)
          {
                    min=c[i][k-1]+c[k+1][j];
                    l=k;
          }
    }
    return l;
}

BTNODE * OBST::construct(int r[10][10],int i,int j)
{
   BTNODE *p;
   if(r[i][j]==0)
             return NULL;
   else
   {
              p=new BTNODE;
              p->word= string(words[r[i][j]]);
              p->left=construct(r,i,r[i][j]-1);
              p->right=construct(r,r[i][j]+1,j);
              return p;
   }
}

void OBST::preorder(BTNODE *T)
{
           if(T!=NULL)
           {
                     cout<<" "<<T->word;
                     preorder(T->left);
                     preorder(T->right);
           }
}

int main()
{
    OBST O;
    O.accept_data();
    O.Optimal_BST();
    O.preorder(O.Root);
          return 0;
}
```