#include <iostream>

using namespace std;

int main()

{

int graph[4][4] =

{

/\*{A,B,C,D}\*/

/\*A\*/{0,2,8,3},

/\*B\*/{2,0,5,1},

/\*C\*/{8,5,0,6},

/\*D\*/{3,1,6,0}

};

int smallest = 500;

int connected[4];

connected[0] = 0;

int n = 0;

int alreadyconnected=0;

int c;

int d;

int totalweightp=0;

cout<<"Prim's algorithm"<<endl;

while(n<3)

{

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

{

int k = connected[b];

for(int i=0; i<4; i++)

{

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

{

alreadyconnected=0;

if(i==connected[m])

{

alreadyconnected=5;

break;

}

}

if(alreadyconnected==5)

continue;

if(graph[k][i]>0 && graph[k][i]< smallest)

{

smallest = graph[k][i];

c = k;

d = i;

}

}

}

n++;

connected[n]=d;

cout<<"element number "<<(c+1)<<" connected to element number "<<(d+1)<<" having weight "<<graph[c][d]<<endl;

totalweightp=totalweightp+graph[c][d];

smallest = 500;

}

cout<<"Total weight of Prim's algorithm = "<<totalweightp<<endl;

cout<<endl<<"Krushkal's algorithm"<<endl;

int totalweightk=0;

int asc[3][6];

cout<<"Weight Source Destination\n";

int l;

int m;

int prevsmallest=0;

for(int e = 0; e<6; e++)

{

for(int f = 0; f<4; f++)

{

for(int g = 0; g<4; g++)

{

if(graph[f][g]<smallest && f!=g && graph[f][g]>prevsmallest)

{

smallest = graph[f][g];

l=f;

m=g;

}

}

}

prevsmallest=smallest;

smallest=500;

asc[0][e]= graph[l][m];

asc[1][e]= l+1;

asc[2][e]= m+1;

cout<<" "<<asc[0][e]<<" "<<asc[1][e]<<" "<<asc[2][e]<<endl;

}

int con[4];

int notcon[4]= {1,2,3,4};

int isin =0;

cout<<"element number "<<asc[1][0]<<" is connected to element number "<<asc[2][0]<<endl;

totalweightk=totalweightk+asc[0][0];

con[0]=asc[1][0];

con[1]=asc[2][0];

cout<<"element number "<<asc[1][1]<<" is connected to element number "<<asc[2][1]<<endl;

totalweightk=totalweightk+asc[0][1];

con[2]=asc[1][1];

con[3]=asc[2][1];

int r;

int s;

int isout=0;

for(int o = 1; o<=4; o++)

{

for(int p =0; p<4; p++)

{

if(o==con[p])

isin=1;

}

if(isin!=1)

{

for(int q=0; q<4;q++)

{

if(graph[o-1][q]<smallest && (o-1)!=q)

smallest = graph[o-1][q];

r= (o-1);

s=q;

}

cout<<"element number "<<(r+1)<<" is connected to element number "<<(s+1)<<endl;

totalweightk=totalweightk+graph[r][s];

isin=0;

isout=1;

}

else

{

isin=0;

}

}

if(isout!=1)

{

cout<<"element number "<<asc[1][2]<<" is connected to element number "<<asc[2][2]<<endl;

totalweightk=totalweightk+asc[0][2];

}

cout<<"Total weight of Kruskal's algorithm = "<<totalweightk<<endl;

return 0;

}