Index

//Normal Prime Seive Algorithm

```
#include <iostream>
#include <vector>
#include <stack>
#include <algorithm>

using namespace std;

#define SIZE_N 100000
#define SIZE_P 66457

bool flag [SIZE_N] ;
int prime [SIZE_P] ;

void seive ()
{
        int i,j,r,c=0,p ;

        for (i=3;i<=SIZE_N;i+=2)    flag[i] = true;

        flag[2]=true;
        prime[++c]=2;
        p=SIZE_N+1;
        for(i=3;i<p;i+=2)
        if ( flag[i] )
        {
            prime[++c]=i;
            if ( SIZE_N / i >= i )
            {
                r=i*2;
                for(j=i*i;j<p;j+=r)
                    flag[j]=false;
            }
        }
        printf( "Total prime: %d\n", c ) ;
}
int main()
{
   seive();
   return 0;
}
```

// Super fast & Memory-tight Sieve by Yarin

```
#include<stdio.h>
#include<string.h>

#define MAXS 10000000
#define MAXSH (MAXS/2)
#define MAXSQ 5000
#define isprime(n) (a[n >> 4] & (1<<(((n)>>1)&7)))
//works when n is odd

char a[MAXS/16+2];

#define PN 5761455
int prime[PN],c;

void seive()
{
        int i,j;
        memset(a,255,sizeof(a));
        a[0]=0xFE;
        for(i=1;i<MAXSQ;i++)
                if (a[i>>3]&(1<<(i&7)))

        for(j=i+i+i+1;j<MAXSH;j+=i+i+1)

        a[j>>3]&=~(1<<(j&7));

        prime[c++]=2;
        for(i=3;i<MAXS;i+=2)
                if(isprime(i))
                        prime[c++]=i;
         printf("Total prime:%d\n",c);
}

int main(){
        seive();
        return 0;
}
```

```
//Compute How many Number Of Divisor Of a Number
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>

using namespace std;

#define SIZE_N 100000
#define SIZE_P 66457

bool flag [SIZE_N] ;
int prime [SIZE_P] ;

void seive ()
{
        int i,j,r,c=0,p ;

        for (i=3;i<=SIZE_N;i+=2)    flag[i] = true;

        flag[2]=true;
        prime[++c]=2;
        p=SIZE_N+1;
        for(i=3;i<p;i+=2)
                if ( flag[i] )
                {
                        prime[++c]=i;
      if ( SIZE_N / i >= i )
      {
         r=i*2;
         for(j=i*i;j<p;j+=r)

        flag[j]=false;
      }
                }
  //printf( "Total prime: %d\n", c ) ;
}

int divisor(int n)
{
  int i,val,count,sum;

  val=sqrt(n)+1;
  sum=1;
  for(i=1;prime[i]<val;i++)
  {
    if(n%prime[i]==0)
    {
      count=0;
      while(n%prime[i]==0)
      {
        n/=prime[i];
        count++;
      }
      sum*=(count+1);
    }
  }
  if(n!=1)
    sum=sum*2;
  return sum;
}

int main()
{
  seive();
  int n;

  while(scanf("%d",&n)==1)
  {
    printf("%d No Of divisor:%d\n",n,divisor(n));
  }
  return 0;
}

/*
Input:
5
8
100
568
Output:
5 No Of divisor:2
8 No Of divisor:4
100 No Of divisor:9
568 No Of divisor:8
*/
```

```
#define  SIZE_N 1000001
#define SIZE_P 100000

int prime[SIZE_P],store[SIZE_N];
long long dp[SIZE_N];
bool flag[SIZE_N];
void sieve()
{
        long i=0,j=0,r=0,c=0,len;
        prime[0]=2;

        for(i=4,j=SIZE_N-1;i<j;i+=2,j-=2)
                flag[i]=flag[j]=1;
        flag[i]=flag[j]=1;

        for(i=3,c=1;i<=SIZE_N;i+=2)
                if(flag[i]==0)
                {
                        prime[c]=i;
                        ++c;
                        if(i<SIZE_N/i)
                        {
                                r=i<<1;

        for(j=i*i;j<=SIZE_N;j+=r)
                                                flag[j]=1;
                        }
                }
}

void r_prime()
{
   int i,j,k,v,count,val;

   dp[1]=0;
   dp[2]=store[1]=store[2]=1;
   dp[3]=3;
   store[3]=2;

   for(i=4;i<SIZE_N;i++)
   {
      if(flag[i]==0)
         store[i]=i-1;
      else
      {
         val=sqrt(i)+1;
for(j=0;prime[j]<val;j++)
            if(i%prime[j]==0)
            {
               count=0;
               v=i;
               while(v%prime[j]==0)
               {
                  v/=prime[j];
                  count++;
               }
               store[i]=(prime[j]-
1)*(pow(prime[j],count-1))*store[v];
               break;
            }
         if(prime[j]>=val)
            store[i]=i-1;
      }
      dp[i]=dp[i-1]+store[i];
   }
}

int main()
{
        sieve();
        r_prime();
        long long q,i,m,temp,j,p,k,t,r;

   //scanf("%lld",&t);
   cin>>t;
   while(t--)
   {
        //scanf("%lld",&r);
        cin>>r;
        //printf("%lld\n",dp[r]);
        cout<<dp[r]<<endl;
   }

        return 0;
}
/*
Input & Output:
10     //test case
25->199        364-> 40335     2154-> 1410815
365->40623   78745-> 1884817298
598-> 108730     813254-> 201035671594  */
```

//GCD Algorithm

```
#include <iostream>
#include <string>
#include <set>
#include <cmath>

using namespace std;

int GCD(int a,int b);

int main()
{
   int a,b,n;

   while(scanf("%d %d",&a,&b)==2)
   {
      n=GCD(a,b);
      printf("%d\n",n);
   }

   return 0;
}

int GCD(int a,int b)
{
   if(b==0)
      return a;
   return GCD(b,a%b);
}
/*
Input & Output:
5 6
1
6 6
6
6 12
6
7 12
1
8 12
4
12 156
12
24 156
12
*/
```

//LCM Algorithm

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <set>
#include <algorithm>
#include <sstream>
#include <cmath>

using namespace std;

int GCD(int a,int b);

int main()
{
   int a,b,n;

   while(scanf("%d %d",&a,&b)==2)
   {
      n=(a*b)/GCD(a,b);
      printf("%d\n",n);
   }

   return 0;
}

int GCD(int a,int b)
{
   if(b==0)
      return a;
   return GCD(b,a%b);
}
/*
Input & Output:
5 6
30
12 56
168
12 14
84
*/
```

```
//Ugly Number Generator
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

#define MAX 2147483647

vector<int> Generate(vector<int>num);

int main()
{
   int n,val,i;
   vector<int>num,dp;

   while(scanf("%d",&n)==1)
   {
      num.clear();
      for(i=0;i<n;i++)
      {
         scanf("%d",&val);
         num.push_back(val);
      }
      dp=Generate(num);
      printf("SIZE:%d %d\n",dp.size(),dp[1499]);
   }
   return 0;
}

vector<int> Generate(vector<int>num)
{
   vector<int>dp;
   int i,j,n,high;
   dp.push_back(1);

   for(i=0;i<num.size();i++)
   {
      high=dp.size();
      n=num[i];
      while(n<MAX)
      {
         for(j=0;j<high;j++)
         {
            if(dp[j]>MAX/n)
               continue;
            dp.push_back(dp[j]*n);
         }
         if(n>MAX/num[i])
            break;
         n=n*num[i];
      }
   }
   sort(dp.begin(),dp.end());
   return dp;
}
/*
Input:
3
2 3 5
Output:
The 1500'th Ugly Number Is:859963392
*/
```

//Big Mod Algorithm

```
#include <iostream>
#include <vector>
using namespace std;

int big_mod(int base,int pow,int mod);

int main()
{
        int b,m,p,result;

        while(scanf("%d %d
%d",&b,&p,&m)==3)
        {
                result=big_mod(b,p,m);
    printf("%d\n",result);
  }
        return 0 ;
}
int big_mod(int base,int pow,int mod)
{
        int temp,result,rem;

        temp=base%mod;
        result=1;

        while(pow>0)
        {
                rem=pow%2;
                pow/=2;

                if(rem==1)
       result=(result*temp)%mod;
                temp=(temp*temp)%mod;
        }
        return result;
}
/*
Input:
2 5 10
2 5 100
Output:
2
32
*/
```

//Xbase To Ybase

```
#include <iostream>
#include <string>
using namespace std;
string BaseConversion(string xstr,int xbase,int ybase);
int main()
{
  string xstr,ystr;
  int xbase,ybase;
  while(cin>>xstr>>xbase>>ybase)
  {
    ystr=BaseConversion(xstr,xbase,ybase);
    cout<<ystr<<endl;
  }
}
string BaseConversion(string xstr,int xbase,int ybase)
{
  string
str="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabc
defghijklmnopqrstuvwxyz";
  long long store[128],i,multi,sum;
  string ystr;

  for(i=sum=0;i<str.size();i++)
    store[str[i]]=i;
  multi=1;
  for(i=xstr.size()-1;i>=0;i--)
  {
    sum+=(store[xstr[i]]*multi);
    multi*=xbase;
  }
  while(true)
  {
    ystr.push_back(str[sum%ybase]);
    sum/=ybase;
    if(sum==0)
      break;
  }
  reverse(ystr.begin(),ystr.end());
  return ystr;
}
/*
Input & Output:
155CBA 16 2      ->10101010110010111010
BCDAE 16 8       ->2746656
abc457 42 9      ->13403018311     */
```

//Extended GCD Algorithm

```
#include<stdio.h>
int lastx,lasty,x,y;

int extended_gcd(int a, int b)
{   int temp , quotient;
        x = 0;     y = 1;
        lasty = 0;  lastx = 1;
        while( b != 0)
        {
                temp = b;
    quotient = a / b;
    b = a % b;
    a = temp;

    temp = x;
    x = lastx - quotient*x;
    lastx = temp;

    temp = y;
    y = lasty - quotient*y;
    lasty = temp;
        }
  return  a;
}

int main()
{       int n,A,B;
        while(scanf("%d%d",&A,&B)==2)
        {
                n=extended_gcd(A,B);
                printf("%d %d %d\n",lastx,lasty,n);
        }
        return 0;
}
```

//Minimum Coin To reach a no.

```
#define INF 1000000
int MinCoin(vector<int>coin,int N);
int main()
{
        int i,n,val;
        vector<int>coin;
        while(scanf("%d",&n)==1)
        {
                coin.clear();
                for(i=0;i<n;i++)
                {
                        scanf("%d",&val);
                        coin.push_back(val);
                }
                scanf("%d",&val);
                val=MinCoin(coin,val);
        cout<<"Minimum Coin:"<<val<<endl;
        }
}
int MinCoin(vector<int>coin,int N)
{
        vector<int>DP(++N,INF);
        int i,j;
        DP[0]=0;
        for(i=0;i<N;i++)
        {
                for(j=0;j<coin.size();j++)
                {
          if(coin[j]<=i && DP[i-coin[j]]+1<DP[i])
                        {
                            DP[i]=DP[i-coin[j]]+1;
                        }
                }
        }
        return DP[N-1];
}
```

Input:

```
3               No. Of Coins
1 3 5           List Of Coins
11              Minimum Coin 3 for 11  (5 5 1)
3
1 3 5
8               Minimum Coin 2 for 8   (5 3)
```

//Coin Change No. Of Ways

```
#include <iostream>
#include <vector>
using namespace std;
int CoinChange(vector<int>coin,int N);
int main()
{
	int i,n,val;
	vector<int>coin;
	while(scanf("%d",&n)==1)
	{
		coin.clear();
		for(i=0;i<n;i++)
		{
			scanf("%d",&val);
			coin.push_back(val);
		}
		scanf("%d",&val);
		val=CoinChange(coin,val);
		cout<<"no.Way:"<<val<<endl;
	}
}

int CoinChange(vector<int>coin,int N)
{
	vector<int>DP(++N,0);
	int i,j;
	DP[0]=1;
	for(i=0;i<coin.size();i++)
	{
		for(j=0;j<N;j++)
		{
			if(j-coin[i]>=0)
			{
			DP[j]+=DP[j-coin[i]];
			}
		}
	}
	return DP[N-1];
}
/*
Input & Output:
3
2 3 5
no.Way:6
*/
```

//Lis Compute In O(n^2)

```
#include <iostream>
#include <vector>
using namespace std;
int LIS(vector<int>data);
int main()
{
	int n,i,val;
	vector<int>Data;
	while(scanf("%d",&n)==1 && n)
	{
		Data.clear();
		for(i=0;i<n;i++)
		{
			scanf("%d",&val);
			Data.push_back(val);
		}
		val=LIS(Data);
		printf("LIS:%d\n",val);
	}
}

int LIS(vector<int>Data)
{
	int i,j,maxv;
	vector<int>DP(Data.size(),1);
	for(i=maxv=1;i<Data.size();i++)
	{
		for(j=i-1;j>=0;j--)
		{
		if(Data[j]<=Data[i] && DP[j]+1>DP[i])
			{
			DP[i]=DP[j]+1;
			maxv=max(maxv,DP[i]);
			}
		}
	}
	return maxv;
}
/*
4
2 3 5 1
LIS:3	(2  3  5)
4
1 4 2 14
LIS:3	(1  4  14)	*/
```

//Longest Common Subsequence(LCS)

```
#include <iostream>
#include <string>
using namespace std;
#define SIZE 100

int LCS(string str1,string str2);
int main()
{
   string str1,str2;
   int len;

   while(cin>>str1>>str2)
   {
      len=LCS(str1,str2);
      printf("%d\n",len);
   }
}
int LCS(string str1,string str2)
{
   int dp[SIZE][SIZE],i,j;

   str1=" "+str1;
   str2=" "+str2;
   for(i=0;i<str1.size();i++)
      dp[i][0]=0;
   for(i=0;i<str2.size();i++)
      dp[0][i]=0;
   for(i=1;i<str1.size();i++)
   {
      for(j=1;j<str2.size();j++)
         if(str1[i]==str2[j])
            dp[i][j]=dp[i-1][j-1]+1;
         else dp[i][j]=max(dp[i][j-1],dp[i-1][j]);
   }
   return dp[i-1][j-1];
}

/*
Input:
BDCABA
ABCBDAB
Output:
lcs:4
*/
```

//MCM(Matrix Chain Multiplication)

```
#define SIZE 100
#define INF 2140000000
int dp[SIZE][SIZE],s[SIZE][SIZE];
void print(int i,int j)
{
        if(i==j)
                printf("A");
        else
        {
                printf("(");
                print(i,s[i][j]);
                print(s[i][j]+1,j);
                printf(")");
        }
}
void MCM(vector<int>p)
{
   int l,j,i,k,q,n;
   n=p.size()-1;
   for(i=1;i<p.size();i++)
      dp[i][i]=0;
   for(l=2;l<=n;l++)
   {
       for(i=1;i<=n-l+1;i++)
       {
          j=i+l-1;
          dp[i][j]=INF;
          for(k=i;k<j;k++)
          {
               q=dp[i][k]+dp[k+1][j]+p[i-1]*p[k]*p[j];
              if(q<dp[i][j])
              {
                    dp[i][j]=q;
                    s[i][j]=k;
              }
          }
       }
    }
   printf("value:%d\n",dp[1][n]);
   print(1,n);
}
/*Input & Output:
7
30 35 15 5 10 20 25
15125  (Min Cost)       ((A(AA))((AA)A))       */
```

//SubSetSum Generate by DP

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
int SubSetSum(vector<int>num);
int main()
{
    int n,val,i;
    vector<int>num;

    while(scanf("%d",&n)==1)
    {
        num.clear();
        for(i=0;i<n;i++)
        {
            scanf("%d",&val);
            num.push_back(val);
        }
        SubSetSum(num);
    }
    return 0;
}
int SubSetSum(vector<int>num)
{
    int i,j,high;
    vector<int>dp;

    dp.push_back(0);
    for(i=0;i<num.size();i++)
    {
        high=dp.size();
        for(j=0;j<high;j++)

dp.push_back(num[i]+dp[j]);
    }
    for(i=0;i<dp.size();i++)
        printf("%d ",dp[i]);
    printf("\n");
}
/*
input & Output:
3
2 3 5
0 2 3 5 5 7 8 10      ->output
*/
```

//Welcome To Google Code Jam Problen in'09

```
char str[]="welcome to code jam";

int google(string str1)
{
        vector<int>dp[510];
        int i,j;
        for(i=0;i<=20;i++)
        {
                for(j=0;j<=str1.size();j++)
                {
                        dp[i].push_back(0);
                }
        }
        for(i=0;i<=str1.size();i++)
                dp[0][i]=1;
        for(i=0;i<19;i++)
        {
                for(j=0;j<str1.size();j++)
                {
                        if(str[i]==str1[j])
                        {

        dp[i+1][j+1]=(dp[i+1][j]+dp[i][j+1])%10000;
                        }
                        else
dp[i+1][j+1]=dp[i+1][j];
                }
        }
        return dp[19][str1.size()];
}
/*
3
elcomew elcome to code jam
wweellccoommee to code qps jam
welcome to codejam
Case #1: 0001
Case #2: 0256
Case #3: 0000
*/
```

//largest Substring length Search

```
int longestMatch(string sequence1,
string sequence2)
{
	int i,j,len1,len2,max;
	vector<int>dp[SIZE];

	len1=sequence1.size();
	len2=sequence2.size();

	for(i=0;i<=len1;i++)
	{
		for(j=0;j<=len2;j++)
		{
			dp[i].push_back(0);
		}
	}

	for(i=max=0;i<len1;i++)
	{
		for(j=0;j<len2;j++)
		{

	if(sequence2[j]==sequence1[i])
			{

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

	if(dp[i+1][j+1]>max)

	max=dp[i+1][j+1];
			}
		}
	}
	return max;
}
```

//Edit Distance

```
#include <iostream>
#include <string>

using namespace std;

int EditDistance(string str1,string str2)
{
  int i,j,cost,dp[100][100];
  str1=" "+str1;
  str2=" "+str2;
  for(i=0;i<str1.size();i++)
    dp[i][0]=i;
  for(j=0;j<str2.size();j++)
    dp[0][j]=j;

  for(i=1;i<str1.size();i++)
    for(j=1;j<str2.size();j++)
    {
      if(str1[i]==str2[j])
        dp[i][j]=dp[i-1][j-1];
      else
      {
        dp[i][j]=min(dp[i-1][j]+1,dp[i][j-1]+1);
        dp[i][j]=min(dp[i][j],dp[i-1][j-1]+1);
      }
    }
  //deletion dp[i-1][j]+1
  //insertion dp[i][j-1]+1
  //substitution dp[i-1][j-1]+1
  return dp[i-1][j-1];
}
/*
kitten
sitting
3
Saturday
Sunday
3
*/
```

```
#include <iostream>
#include <string>
#include <map>
#include <vector>
#include <sstream>
#include <algorithm>
#include <stack>
#include <queue>
#include <cmath>

using namespace std;

#define SIZE 100000
#define INF 1000000
#define NILL -1
#define white 1                 //white: Undiscovered
#define gray 2                  //gray: Discovered
#define black 3                 //black: Finished

vector<int> BFS(vector<int>adj[],int source,int nodes);

int main()
{
    int i,u,v,nodes,edges,source;
    vector<int>adj[SIZE];

    while(scanf("%d %d",&nodes,&edges)==2)
    {
        vector<int>dis(nodes);

        for(i=0;i<nodes;i++)
            adj[i].clear();
        for(i=0;i<nodes;i++)
        {
            scanf("%d %d",&u,&v);
            adj[u].push_back(v);
        }
        scanf("%d",&source);
        dis=BFS(adj,source,nodes);

        for(i=0;i<nodes;i++)
        {
            if(dis[i]!=INF)
                printf("Dis node %d From %d : %d\n",i,source,dis[i]);
            else printf("Dis node %d From %d : INF\n",i,source);
        }
    }
    return 0;
}

vector<int> BFS(vector<int>adj[],int source,int nodes)
{
    vector<int>dis(nodes,INF),par(nodes,NILL),color(nodes,white);
    queue<int>Q;
```

```
    int i,u,v;

    dis[source]=0;
    par[source]=NILL;
    Q.push(source);
    color[source]=gray;

    while(!Q.empty())
    {
       u=Q.front();
       Q.pop();
       for(i=0;i<adj[u].size();i++)
       {
           v=adj[u][i];
           if(color[v]==white)
           {
               color[v]=gray;
               dis[v]=dis[u]+1;
               par[v]=u;
               Q.push(v);
           }
           color[u]=black;
       }
    }
    return dis;
}

/*
    Input:
    4 4
    0 1
    0 2
    2 1
    1 3
    0
    Output:
    Dis node 0 From 0 : 0
    Dis node 1 From 0 : 1
    Dis node 2 From 0 : 1
    Dis node 3 From 0 : 2
    Input:
    4 4
    0 1
    0 2
    2 1
    1 3
    2
    Output:
    Dis node 0 From 2 : INF
    Dis node 1 From 2 : 1
    Dis node 2 From 2 : 0
    Dis node 3 From 2 : 2
*/
```

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <algorithm>
#include <stack>
#include <queue>
#include <sstream>

using namespace std;

#define SIZE 100000
#define NILL -1
#define white 1
#define gray 2
#define black 3


vector<int>Dis(SIZE),Par(SIZE),Color(SIZE),Fin(SIZE),adj[SIZE];
int Time;

int DFS(int nodes);
void DFS_Visit(int u);

int main()
{
    int nodes,edges,i,u,v;

    while(scanf("%d %d",&nodes,&edges)==2)
    {
        for(i=0;i<nodes;i++)
            adj[i].clear();
        for(i=0;i<edges;i++)
        {
            scanf("%d %d",&u,&v);
            adj[u].push_back(v);
        }
        DFS(nodes);
        for(i=0;i<nodes;i++)
            printf("%d ",Par[i]);
        printf("\n");
    }

    return 0;
}

int DFS(int nodes)
{
    int i;

    for(i=0;i<nodes;i++)
    {
        Color[i]=white;
        Par[i]=NILL;
```

```
    }
    Time=0;
    for(i=0;i<nodes;i++)
        if(Color[i]==white)
            DFS_Visit(i);
}

void DFS_Visit(int u)
{
    Color[u]=gray;      //White Vertex u Has just been Discovered
    Dis[u]=++Time;        //Discoverd u Vertex
    for(int i=0;i<adj[u].size();i++)
        if(Color[adj[u][i]]==white)
        {
            Par[adj[u][i]]=u;
            DFS_Visit(adj[u][i]);
        }
    Color[u]=black;     //Blacken u,it is finished
    Fin[u]=++Time;
}
```

```
#include <vector>
#include <list>

using namespace std;

#define SIZE 100
#define INF 100000

struct pq
{
	int cost,node;
	bool operator<(const pq &b)const
	{
		return cost>b.cost;	// Min Priority Queue
	}
};

vector<pq>adj[SIZE];

vector<int> Dijkstra(int source,int nodes);

int main()
{
	int nodes,edges,i,u,v,cost,source;
	pq V;
	vector<int>dist;

	while(scanf("%d %d",&nodes,&edges)==2)
	{
		for(i=0;i<nodes;i++)
		{
			adj[i].clear(); //clear adj vector
		}
		for(i=0;i<edges;i++)
		{
			scanf("%d %d %d",&u,&v,&cost);
			V.cost=cost;
			V.node=v;
			adj[u].push_back(V);
			/*V.node=u;			//For Bidirectional Edges
			adj[v].push_back(V);*/
		}
		scanf("%d",&source);
		dist=Dijkstra(source,nodes);
		for(i=0;i<nodes;i++)
		{
			cout<<dist[i]<<" ";
		}
		cout<<endl;
	}

	return 0;
}
```

```
vector<int> Dijkstra(int source,int nodes)
{
      priority_queue<pq>Q;
      vector<int>dist;
      pq U,V;
      int i;

      for(i=0;i<nodes;i++)
      {
            dist.push_back(INF);
      }
      dist[source]=0;
      V.node=source;
      V.cost=0;
      Q.push(V);
      while(!Q.empty())
      {
            U=Q.top();
            Q.pop();
            for(i=0;i<adj[U.node].size();i++)
            {

      if(dist[U.node]+adj[U.node][i].cost<dist[adj[U.node][i].node])
                  {

      dist[adj[U.node][i].node]=dist[U.node]+adj[U.node][i].cost;
                        V.node=adj[U.node][i].node;
                        V.cost=dist[adj[U.node][i].node];
                        Q.push(V);
                  }
            }
      }
      return dist;
}

/*
Input:
6 9
0 1 4
0 2 2
1 2 1
1 3 5
2 3 8
2 4 10
3 5 6
3 4 2
4 5 3
0
Output:

Shortest Path All the Node Frome Source:0
0 3 2 8 10 13
*/
```

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <stack>
#include <queue>
#include <sstream>
#include <cmath>

using namespace std;

#define SIZE 100000
#define INF 1000000
#define NILL -1
#define white 1
#define gray 2
#define black 3

typedef struct
{
    int Time,u;
}Node;

vector<int>adj[SIZE],adjT[SIZE],Dis(SIZE),Par(SIZE),Fin(SIZE),Color(SIZE),ans
[SIZE];
int Time,K;

void StronglyConnectedComponents(int nodes);
void TransposeAdjList(int nodes);
int TDFS(int nodes);
int DFS(int nodes);
void DFS_Visit(int u);
void TDFS_Visit(int u);
int comp(Node P,Node Q);

int main()
{
    int nodes,edges,u,v,i,j,Count;

    while(scanf("%d %d",&nodes,&edges)==2)
    {
        for(i=0;i<nodes;i++)
        {
            adj[i].clear();
            adjT[i].clear();
            ans[i].clear();
        }
        for(i=0;i<edges;i++)
        {
            scanf("%d %d",&u,&v);
            adj[u].push_back(v);
        }
        StronglyConnectedComponents(nodes);
        printf("Strongly Connected Components Groups:\n");
```

```
        Count=1;
        for(i=0;i<K;i++)
        {
            printf("Group No:%d\n",Count++);
            for(j=0;j<ans[i].size();j++)
                printf("%d ",ans[i][j]);
            printf("\n");
        }
    }
    return 0;
}

void StronglyConnectedComponents(int nodes)
{
    DFS(nodes);
    TransposeAdjList(nodes);
    TDFS(nodes);
}

int DFS(int nodes)
{
    int i;

    for(i=0;i<nodes;i++)
    {
        Color[i]=white;
        Par[i]=NILL;
    }
    Time=0;
    for(i=0;i<nodes;i++)
        if(Color[i]==white)
            DFS_Visit(i);
}

void DFS_Visit(int u)
{
    Color[u]=gray;      //White Vertex u Has just been Discovered
    Dis[u]=++Time;        //Discoverd u Vertex
    for(int i=0;i<adj[u].size();i++)
        if(Color[adj[u][i]]==white)
        {
            Par[adj[u][i]]=u;
            DFS_Visit(adj[u][i]);
        }
    Color[u]=black;     //Blacken u,it is finished
    Fin[u]=++Time;
}

void TransposeAdjList(int nodes)          //Transpose Adjacency List
{
    int i,j;

    for(i=0;i<nodes;i++)
        for(j=0;j<adj[i].size();j++)
            adjT[adj[i][j]].push_back(i);
}
```

```
int comp(Node P,Node Q)
{
    return P.Time>Q.Time;
}

int TDFS(int nodes)     // DFS for Transpose Adj List
{
    int i;

    Node P;
    vector<Node>Array(nodes);       //Sort For Finishing Times
for(i=0;i<nodes;i++)
    {
        P.Time=Fin[i];
        P.u=i;
        Array[i]=P;
    }
    sort(&Array[0],&Array[i],comp);
    for(i=0;i<nodes;i++)
    {
        Color[i]=white;
        Par[i]=NILL;
    }
    Time=0;
    for(i=K=0;i<nodes;i++)
        if(Color[Array[i].u]==white)
        {
            TDFS_Visit(Array[i].u);
            K++;        //Group No Increase
        }
}
void TDFS_Visit(int u)
{
    Color[u]=gray;      //White Vertex u Has just been Discovered
    Dis[u]=++Time;        //Discoverd u Vertex
    for(int i=0;i<adjT[u].size();i++)
        if(Color[adjT[u][i]]==white)
        {
            Par[adjT[u][i]]=u;
            TDFS_Visit(adjT[u][i]);       //Be CareFull Function Call
        }
    Color[u]=black;     //Blacken u,it is finished
    ans[K].push_back(u);      //Store Member of a group
    Fin[u]=++Time;
}
/*
Input:
8 14
0 1
1 2
1 5
1 4
2 6
2 3
3 2
3 7
4 5
```

```
5 6
7 6
7 3
6 5
4 0

Output:

Strongly Connected Components Groups:
Group No:1
1 4 0
Group No:2
7 3 2
Group No:3
5 6

Input:
8 14
0 1
1 2
1 5
1 4
2 6
2 3
3 2
3 7
4 0
4 5
5 6
6 5
6 7
7 7
Output:
Strongly Connected Components Groups:
Group No:1
1 4 0
Group No:2
3 2
Group No:3
5 6
Group No:4
7
*/
```

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <map>
#include <stack>
#include <queue>
#include <sstream>

using namespace std;

#define SIZE 10000
#define NILL -1
#define white 1
#define gray 2
#define black 3

vector<int>Dis(SIZE),Par(SIZE),Color(SIZE),adj[SIZE],dfsTree[SIZE],Low(SIZE);
int Time;

vector<int> ArticulationPoint(int nodes,int source);
void DFS_Visit(int u);

int main()
{
    int nodes,edges,i,u,v;
    vector<int>Point;

    while(scanf("%d %d",&nodes,&edges)==2)
    {
        for(i=0;i<nodes;i++)
        {
            adj[i].clear();
            dfsTree[i].clear();
        }

        for(i=0;i<edges;i++)
        {
            scanf("%d %d",&u,&v);
            adj[u].push_back(v);
            adj[v].push_back(u);
        }
        Point=ArticulationPoint(nodes,0);
        if(Point.size())
        {
            for(i=0;i<Point.size();i++)
                printf("%d is articulation point.\n",Point[i]);
        }
        else printf("No articulation Point.\n");
    }
    return 0;
}

vector<int> ArticulationPoint(int nodes,int source)
```

```
{
    int i,u,v;
    vector<int>Point;

    for(i=0;i<nodes;i++)        //Initialize
    {
        Color[i]=white;
        Dis[i]=0;
        Par[i]=NILL;
    }
    Time=0;
    DFS_Visit(source);
    for(u=0;u<nodes;u++)            //Check articulation Point
    {
        if(u==source)
            continue;
        for(i=0;i<dfsTree[u].size();i++)
        {
            v=dfsTree[u][i];
            if(Low[v]>=Dis[u])
            {
                Point.push_back(u);
                break;
            }
        }
    }
    return Point;
}

void DFS_Visit(int u)
{
    Color[u]=gray;      //White Vertex u Has just been Discovered
    Dis[u]=Low[u]=++Time;      //Discoverd u Vertex
    for(int i=0;i<adj[u].size();i++)
        if(Color[adj[u][i]]==white)
        {
            Par[adj[u][i]]=u;
            dfsTree[u].push_back(adj[u][i]);
            DFS_Visit(adj[u][i]);
            Low[u]=min(Low[u],Low[adj[u][i]]);
        }
        else if(adj[u][i]!=Par[u])
            Low[u]=min(Low[u],Dis[adj[u][i]]);
    Color[u]=black;     //Blacken u,it is finished
}


/*
Input:
4 3
0 1
1 2
2 3
Output:
1 is articulation point.
2 is articulation point.
Input:
```

```
23 32
0 1
1 6
0 2
2 6
0 6
1 2
6 5
3 4
4 5
3 5
5 7
5 8
8 7
8 9
5 10
10 11
11 13
10 12
12 13
13 14
14 17
17 15
17 16
16 15
15 18
17 18
18 19
18 20
20 22
22 21
21 20
14 18
Output:
5 is articulation point.
6 is articulation point.
8 is articulation point.
10 is articulation point.
13 is articulation point.
14 is articulation point.
18 is articulation point.
20 is articulation point.
*/
```

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <stack>
#include <queue>
#include <sstream>
#include <cmath>

using namespace std;

#define INF 1000000
#define NILL -1

typedef struct
{
    int u,v,cost;
}Node;

int BellManFord(vector<Node>Edge,int nodes,int source,int end);

int main()
{
    int nodes,edges,u,v,cost,i,source,end,val;
    vector<Node>Edge;

    while(scanf("%d %d",&nodes,&edges)==2)
    {
        Edge.clear();       //Clear Edge Vector
        Node P;
        for(i=0;i<edges;i++)
        {
            scanf("%d %d %d",&u,&v,&cost);
            P.u=u;
            P.v=v;
            P.cost=cost;
            Edge.push_back(P);
            P.v=u;      //For Bidirectional Edges
            P.u=v;
            Edge.push_back(P);
        }
        scanf("%d %d",&source,&end);
        val=BellManFord(Edge,nodes,source,end);
        if(val==-1)
            printf("Unreachable\n");
        else printf("Shortest Cost:%d\n",val);
    }

    return 0;
}

int BellManFord(vector<Node>Edge,int nodes,int source,int end)
{
    vector<int>Dis(nodes,INF),Par(nodes,NILL);
```

```
    int i,j;

    Dis[source]=0;
    for(i=0;i<nodes-1;i++)
    {
        for(j=0;j<Edge.size();j++)
        {
            if(Dis[Edge[j].v]>Dis[Edge[j].u]+Edge[j].cost)
            {
                Dis[Edge[j].v]=Dis[Edge[j].u]+Edge[j].cost;
                Par[Edge[j].v]=Edge[j].u;
            }
        }
    }
    for(i=0;i<Edge.size();i++)      //Negative Edge Cycle Detection
        if(Dis[Edge[i].v]>Dis[Edge[i].u]+Edge[i].cost)
            return -1;          //Return -1 Because Unreachable Infinity
Solution

    if(Dis[end]==INF)
        return -1;         //If There Is no Negative Cycle Then Dis Vector
Array
    return Dis[end];      //act as Dijkstra. all shortest  Distance From
Source

}
/*
Input:
6 9
0 1 4
0 2 2
1 2 1
1 3 5
2 3 8
2 4 10
3 5 6
3 4 2
4 5 3
0 5         //source & destination
Output:
13
Input:
6 6
0 1 5
0 3 2
1 2 -2
3 4 3
4 5 1
2 5 2
0 5
Output:
Unreachable
*/
```

```
//Total Complexit n^2(logn)

#include <vector>
#include <map>
#include <set>
#include <deque>
#include <stack>
#include <algorithm>
#include <sstream>
#include <iostream>
#include <cstdio>
#include <cmath>
#include <queue>

using namespace std;

#define SIZE 100
#define INF 100000

struct pq
{
	int cost,node;
	bool operator<(const pq &b)const
	{
		return cost>b.cost; // Min Priority Queue
	}
};

vector<pq>adj[SIZE];

vector<int> Dijkstra(int source,int nodes);

int main()
{
	int nodes,edges,i,u,v,cost,source,destination;
	pq V;
	vector<int>dp[SIZE];

	while(scanf("%d %d",&nodes,&edges)==2)
	{
		for(i=0;i<nodes;i++)
		{
			adj[i].clear(); //clear adj vector
			dp[i].clear();
		}
		for(i=0;i<edges;i++)
		{
			scanf("%d %d %d",&u,&v,&cost);
			V.cost=cost;
			V.node=v;
			adj[u].push_back(V);
			V.node=u;                    //For Bidirectional Edges
			adj[v].push_back(V);
		}
```

```
            for(i=0;i<nodes;i++)
            {
                  dp[i]=Dijkstra(i,nodes);
            }
            for(i=0;i<nodes;i++)
            {
            for(int j=0;j<nodes;j++)
            {
                printf("%7d ",dp[i][j]);
            }
            cout<<endl;
        }
            scanf("%d %d",&source,&destination);
            cout<<"Cost:"<<dp[source][destination]<<endl;  // Not Possible
When Value==INF
      }

      return 0;
}

vector<int> Dijkstra(int source,int nodes)
{
      priority_queue<pq>Q;
      vector<int>dist;
      pq U,V;
      int i;

      for(i=0;i<nodes;i++)
      {
            dist.push_back(INF);
      }
      dist[source]=0;
      V.node=source;
      V.cost=0;
      Q.push(V);
      while(!Q.empty())
      {
            U=Q.top();
            Q.pop();
            for(i=0;i<adj[U.node].size();i++)
            {

      if(dist[U.node]+adj[U.node][i].cost<dist[adj[U.node][i].node])
                  {

      dist[adj[U.node][i].node]=dist[U.node]+adj[U.node][i].cost;
                        V.node=adj[U.node][i].node;
                        V.cost=dist[adj[U.node][i].node];
                        Q.push(V);
                  }
            }
      }
      return dist;
}
```

```
/*Floyd warshall algorithm by chormen*/

#define NIL 0
#define sz 10
#define INF 100000

long mat[sz][sz],path[sz][sz], warshall[sz][sz];

long vertex,edges;

void PrintWarshall(long print[][sz])
{
      long ind, jnd;
      printf("All Pairs Shortest Path : \n\n");

      for(ind=1;ind<=vertex; ind++)
      {
            for(jnd=1;jnd<=vertex; jnd++)
            {
                  printf("%ld\t",print[ind][jnd]);
            }
            printf("\n");
      }

}
void PrintAllPairsShortestPath(long ind, long jnd)
{
      if(ind==jnd)
            printf("%ld",ind);
      else
      {
            if(path[ind][jnd]==NIL)
            {
                  printf("NO Path From I to J exists.\n");
            }
            else
            {
                        PrintAllPairsShortestPath(ind, path[ind][jnd]);
                        printf("->%ld",jnd);
            }
      }
}
void FloydWarshall()
{
      long ind,jnd,knd,value;

      for(ind=1; ind<=vertex; ind++)
            for(jnd=1; jnd<=vertex;jnd++)
            {
                  if(ind!=jnd)
                        warshall[ind][jnd]=mat[ind][jnd];
                  else
                        warshall[ind][jnd]=0;
```

```
                  if(ind==jnd || warshall[ind][jnd]>=INF)
                        path[ind][jnd]=NIL;
                  else
                        path[ind][jnd]=ind;

            }

      for(knd=1; knd<=vertex; knd++)
            for(ind=1; ind<=vertex; ind++)
                  for(jnd=1; jnd<=vertex; jnd++)
                  {
                        value=warshall[ind][knd]+warshall[knd][jnd];
                        if(warshall[ind][jnd]>value)
                        {
                              warshall[ind][jnd]=value;
                              path[ind][jnd]=path[knd][jnd];
                        }
                  }
}

int main(void)
{
      long ind,jnd,kase=0;
      long row,col,cost;
      while(scanf("%ld",&vertex)==1)
      {
            scanf("%ld",&edges);
            for(ind=1; ind<=vertex; ind++)
                  for(jnd=1; jnd<=vertex;jnd++)
                        mat[ind][jnd]=INF;
            for(ind=1; ind<=edges; ind++)
            {
                  scanf("%ld %ld",&row,&col);
                  scanf("%ld",&cost);

                  mat[row][col]=cost;
            }
            printf("Graph %ld : The Calcultaion From The Floyd Warshall
:\n",++kase);
            FloydWarshall();
            PrintWarshall(warshall);
            printf("The path from node ind to jnd\n\n");
            PrintWarshall(path);
            for(ind=1; ind<=vertex; ind++)
                  for(jnd=1; jnd<=vertex; jnd++)
                  {
                        printf("Thepath form node %ld to %ld :\n",ind,jnd);
                        PrintAllPairsShortestPath(ind,jnd);
                        printf("\n");
                  }
      }
}
```

```
#include <vector>
#include <list>
#include <map>
#include <set>
#include <deque>
#include <stack>
#include <bitset>
#include <algorithm>
#include <functional>
#include <numeric>
#include <utility>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <queue>

using namespace std;

#define SIZE 10000

struct pq
{
	int cost,node;
	bool operator<(const pq &b)const
	{
		return cost>b.cost;	// Min Priority Queue
	}
};

int Prims(vector<pq>adj[],int source,int nodes);

int main()
{
	int nodes,edges,i,u,v,cost,source,val;
	pq V;
	vector<int>dist;
	vector<pq>adj[SIZE];

	while(scanf("%d %d",&nodes,&edges)==2)
	{
		for(i=0;i<nodes;i++)
		{
			adj[i].clear(); //clear adj vector
		}
		for(i=0;i<edges;i++)
		{
			scanf("%d %d %d",&u,&v,&cost);
			V.cost=cost;
			V.node=v;
			adj[u].push_back(V);
```

```
                  V.node=u;                          //For Bidirectional Edges
                  adj[v].push_back(V);
            }
            val=Prims(adj,0,nodes);
            printf("%d\n",val);
      }

      return 0;
}

int Prims(vector<pq>adj[],int source,int nodes)
{
      priority_queue<pq>Q;
      vector<int>color(nodes);
      pq U,V;
      int i,sum;

      V.node=source;
      sum=V.cost=0;
      Q.push(V);
      while(!Q.empty())
      {
            U=Q.top();
            if(color[U.node]==0)
            sum+=U.cost;
            color[U.node]=1;
            Q.pop();
            for(i=0;i<adj[U.node].size();i++)
            {
                  if(color[ adj[U.node][i].node]==0)
                  {
                        V.node=adj[U.node][i].node;
                        V.cost=adj[U.node][i].cost;
                        Q.push(V);
                  }
            }
      }
      return sum;
}
/*
    Input:
    5 7
    0 1 5
    0 2 4
    1 2 2
    1 4 3
    2 4 1
    2 3 4
    3 4 3
    Output:
    10
*/
```

```
#define SIZE 10000

typedef struct
{
    int u,v,cost;
}Node;

vector<int>Par(SIZE),Rank(SIZE);

int Kruskal(vector<Node>Edge,int nodes);
int comp(Node P,Node Q);
void MakeSet(int nodes);
int FindSet(int x);
void Union(int x,int y);
void Link(int x,int y);

int main()
{
    int i,u,v,nodes,edges,cost,val;
    vector<Node>Edge;

    while(scanf("%d %d",&nodes,&edges)==2)
    {
        Node P;
        Edge.clear();
        for(i=0;i<edges;i++)
        {
            scanf("%d %d %d",&u,&v,&cost);
            P.u=u;
            P.v=v;
            P.cost=cost;
            Edge.push_back(P);
        }
        val=Kruskal(Edge,nodes);
        printf("%d\n",val);
    }
    return 0;
}
int Kruskal(vector<Node>Edge,int nodes)
{
    int i,sum;
    sort(Edge.begin(),Edge.end(),comp);
    MakeSet(nodes);
    for(i=sum=0;i<Edge.size();i++)
    {
        if(FindSet(Edge[i].u)!=FindSet(Edge[i].v))
        {
            Union(Edge[i].u,Edge[i].v);
            sum+=Edge[i].cost;
        }
    }
    return sum;
}
```

```
void MakeSet(int nodes)
{
    for(int i=0;i<nodes;i++)
    {
        Par[i]=i;
        Rank[i]=i;
    }
}

int FindSet(int x)
{
    if(x!=Par[x])
        Par[x]=FindSet(Par[x]);
    return Par[x];
}

void Union(int x,int y)
{
    Link(FindSet(x),FindSet(y));
}

void Link(int x,int y)
{
    if(Rank[x]>Rank[y])
        Par[y]=x;
    else
    {
        Par[x]=y;
        if(Rank[x]==Rank[y])
            Rank[y]=Rank[y]+1;
    }
}

int comp(Node P,Node Q)
{
    if(P.cost > Q.cost)
        return false;
    return true;
}

/*
    Input:
    5 7
    0 1 5
    0 2 4
    1 2 2
    1 4 3
    2 4 1
    2 3 4
    3 4 3
    Output:
    10
*/
```

```
#include <iostream>
#include <map>
#include <string>
#include <vector>
#include <algorithm>
#include <stack>
#include <queue>

using namespace std;

#define SIZE 10000
#define NILL -1
#define white 1
#define gray 2
#define black 3

vector<int>adj[SIZE],Dis(SIZE),Par(SIZE),Array(SIZE),Color(SIZE),Fin(SIZE),In
deg(SIZE),Rank(SIZE);
int Time,K;

int TopologicalSort(int nodes);
int DFS(int nodes);
void DFS_Visit(int u);

int main()
{
    int i,j,nodes,edges,u,v,val;

    while(scanf("%d %d",&nodes,&edges)==2)
    {
        for(i=0;i<nodes;i++)
        {
            adj[i].clear();
            Indeg[i]=0;
            Rank[i]=1;
        }
        for(i=0;i<edges;i++)
        {
            scanf("%d %d",&u,&v);
            adj[u].push_back(v);
            Indeg[v]++;       //Indegree count
        }
        val=TopologicalSort(nodes);
        if(val)
        {
            for(i=0;i<K;i++)
                printf("%d ",Array[i]);
            printf("\n");
        }
        else printf("Impossible\n");

    }
    return 0;
}
```

```
int TopologicalSort(int nodes)
{
    int i,j;

    DFS(nodes);
    reverse(&Array[0],&Array[K]);
    for(i=0;i<K;i++)
    {
        if(Indeg[Array[i]]) //Impossible Case Check
            return 0;
        for(j=0;j<adj[Array[i]].size();j++)
        {
            Indeg[adj[Array[i]][j]]--;
            if(Rank[adj[Array[i]][j]]<=Rank[Array[i]])
                Rank[adj[Array[i]][j]]=Rank[Array[i]]+1;
        }
    }
    return 1;
}

int DFS(int nodes)
{
    int i;

    for(i=0;i<nodes;i++)
    {
        Color[i]=white;
        Par[i]=NILL;
    }
    Time=K=0;
    for(i=0;i<nodes;i++)
        if(Color[i]==white)
            DFS_Visit(i);
}

void DFS_Visit(int u)
{
    Color[u]=gray;      //White Vertex u Has just been Discovered
    Dis[u]=++Time;        //Discoverd u Vertex
    for(int i=0;i<adj[u].size();i++)
        if(Color[adj[u][i]]==white)
        {
            Par[adj[u][i]]=u;
            DFS_Visit(adj[u][i]);
        }
    Color[u]=black;     //Blacken u,it is finished
    Fin[u]=++Time;
    Array[K++]=u;       //Store the node topological sorted order but in
reversely
}

/*
Input:
4 4
0 1
0 2
```

```
1 2
2 3
Output:
0 1 2 3
Input:
9 10
0 1
2 1
3 1
4 1
2 3
3 4
5 6
6 7
4 7
5 4
Output:
8 5 6 2 3 4 7 0 1
Input:
2 2
0 1
1 0
Output:
Impossible

*/
```

```
#include<math.h>
#include<stdio.h>
#include <string.h>

#define M 128
#define N 128

int graph[M][N],edg[M];
bool seen[N];
int matchL[M], matchR[N];
int n, m;
double gop[110][2],hol[110][2];

void input(long d)
{
      long i,j;
      double dis,x,y;
      for(i=0;i<n;i++)
            scanf("%lf %lf",&gop[i][0],&gop[i][1]);
      for(i=0;i<m;i++)
            scanf("%lf %lf",&hol[i][0],&hol[i][1]);
      for(i=0;i<n;i++)
      {
            for(j=0;j<m;j++)
            {
                  x=gop[i][0]-hol[j][0];
                  y=gop[i][1]-hol[j][1];
                  dis=sqrt(x*x+y*y);
                  if(d>dis||(dis-d)<=1e-10)
                  {
                        graph[j][edg[j]++]=i;
                  }
            }
      }
}

bool bpm( int u )
{
      int i,v;
    for( i = 0; i < edg[u]; i++ )
      {
            v=graph[u][i];
            if( seen[v] ) continue;
            seen[v] = true;

            if( matchR[v] < 0 || bpm( matchR[v] ) )
            {
                  matchL[u] = v;
                  matchR[v] = u;
                  return true;
            }
      }
    return false;
}
```

```
int main()
{
      //freopen("bpm.txt","r",stdin);
      long e,i,s,d;
    // Read input and populate graph[][]
    // Set m, n
      while(4==scanf("%d %d %ld %ld",&n,&m,&s,&d))
      {

            e=(n>m)?n:m;
            memset( edg, 0, sizeof(int ) *e );
            input(s*d);

            memset( matchL, -1, sizeof( int ) *m );
            memset( matchR, -1, sizeof( int ) *n );
            int cnt = 0;
            for( i = 0; i < m; i++ )
            {
                  memset( seen, 0, sizeof( seen ) );
                  if( bpm( i ) ) cnt++;
            }
            printf("%d\n",n-cnt);
      }

    // cnt contains the number of happy pigeons
    // matchL[i] contains the hole of pigeon i or -1 if pigeon i is unhappy
    // matchR[j] contains the pigeon in hole j or -1 if hole j is empty

    return 0;
}
```

/* Input: 2 2 5 10 1.0 1.0 2.0 2.0 100.0 100.0 20.0 20.0 Output: 1 */	The gopher family, having averted the canine threat, must face a new predator. The are *n* gophers and *m* gopher holes, each at distinct (x, y) coordinates. A hawk arrives and if a gopher does not reach a hole in *s* seconds it is vulnerable to being eaten. A hole can save at most one gopher. All the gophers run at the same velocity *v*. The gopher family needs an escape strategy that minimizes the number of vulnerable gophers. The input contains several cases. The first line of each case contains four positive integers less than 100: *n*, *m*, *s*, and*v*. The next *n* lines give the coordinates of the gophers; the following *m* lines give the coordinates of the gopher holes. All distances are in metres; all times are in seconds; all velocities are in metres per second. Output consists of a single line for each case, giving the number of vulnerable gophers.

```
#include <iostream>
#include <string>
using namespace std;
string Addition(string str1,string str2);
int main()
{
    string str1,str2,str;
    char s1[100],s2[100];
    int i;

    while(scanf("%s %s",s1,s2)==2)
    {
        str1.clear();
        for(i=0;s1[i]!='\0';i++)
            str1.push_back(s1[i]);
        str2.clear();
        for(i=0;s2[i]!='\0';i++)
            str2.push_back(s2[i]);
        str=Addition(str1,str2);
        cout<<str<<endl;
    }
    return 0;
}

string Addition(string str1,string str2)
{
    int i,j,carry,sum;
    string str;

    for(carry=0,i=str1.size()-1,j=str2.size()-1;i>=0 && j>=0;i--,j--)
    {
        sum=(str1[i]-'0')+(str2[j]-'0')+carry;
        carry=sum/10;
        str.push_back(sum%10+'0');
    }
    for(i=i;i>=0;i--)
    {
        sum=(str1[i]-'0')+carry;
        carry=sum/10;
        str.push_back(sum%10+'0');
    }
    for(j=j;j>=0;j--)
    {
        sum=(str2[j]-'0')+carry;
        carry=sum/10;
        str.push_back(sum%10+'0');
    }
    if(carry)
        str.push_back(carry+'0');
    reverse(str.begin(),str.end());
    return str;
}
```

```
#include <iostream>
#include <string>
using namespace std;
string Multiplication(string str1,string str2);
string Addition(string str1,string str2);

int main()
{
    char s1[100],s2[100];
    string str1,str2,str;
    int i;

    while(scanf("%s %s",s1,s2)==2)
    {
        str1.clear();
        for(i=0;s1[i]!='\0';i++)
            str1.push_back(s1[i]);
        str2.clear();
        for(i=0;s2[i]!='\0';i++)
            str2.push_back(s2[i]);
        str=Multiplication(str1,str2);
        cout<<str<<endl;
    }

    return 0;
}

string Multiplication(string str1,string str2)
{
    int i,j,multi,carry;
    string str,temp;

    str="0";
    for(j=str2.size()-1;j>=0;j--)
    {
        temp.clear();
        carry=0;
        for(i=str1.size()-1;i>=0;i--)
        {
            multi=(str1[i]-'0')*(str2[j]-'0')+carry;
            temp.push_back(multi%10+'0');
            carry=multi/10;
        }
        if(carry)
            temp.push_back(carry+'0');
        reverse(temp.begin(),temp.end());
        for(i=j+1;i<str2.size();i++)
            temp.push_back('0');
        str=Addition(str,temp);
    }
    return str;
}
```

```
#include <iostream>

using namespace std;

string Multiplication(string str1,string str2);
string Addition(string str1,string str2);

int main()
{
    char s1[100],s2[100];
    string str1,str2,str;
    int i;

    while(scanf("%s %s",s1,s2)==2)
    {
        str1.clear();
        for(i=0;s1[i]!='\0';i++)
            str1.push_back(s1[i]);
        str2.clear();
        for(i=0;s2[i]!='\0';i++)
            str2.push_back(s2[i]);
        str=Multiplication(str1,str2);
        cout<<str<<endl;
    }

    return 0;
}

string Multiplication(string str1,string str2)
{
    int i,j,multi,carry;
    string str,temp;

    str="0";
    for(j=str2.size()-1;j>=0;j--)
    {
        temp.clear();
        carry=0;
        for(i=str1.size()-1;i>=0;i--)
        {
            multi=(str1[i]-'0')*(str2[j]-'0')+carry;
            temp.push_back(multi%10+'0');
            carry=multi/10;
        }
        if(carry)
            temp.push_back(carry+'0');
        reverse(temp.begin(),temp.end());
        for(i=j+1;i<str2.size();i++)
            temp.push_back('0');
        str=Addition(str,temp);
    }
    return str;
}
```

```
int post_eval(string post);
int main()
{
    string post;
    int val;

    while(cin>>post)
    {
        val=post_eval(post);
        cout<<val<<endl;
    }
}
int post_eval(string post)
{
    stack<int>stk;
    int i,val1,val2;

    for(i=0;i<post.size();i++)
    {
        if(isdigit(post[i]))
            stk.push(post[i]-'0');
        else
        {
            val2=stk.top();
            stk.pop();
            val1=stk.top();
            stk.pop();
            if(post[i]=='+')
            {
                stk.push(val1+val2);
            }
            if(post[i]=='-')
            {
                stk.push(val1-val2);
            }
            if(post[i]=='*')
            {
                stk.push(val1*val2);
            }
            if(post[i]=='/')
            {
                stk.push(val1/val2);
            }
        }
    }
    return stk.top();
}
/*
    Input:
    562+*84/-
    2556+*+
    Output:
    38
    57    */
```

```
#include <stdio.h>
#include <string.h>

int post(char in[],char root,int n);

char par[1000];

int main()
{
    int test,i,j,k,l,n;
    char pre[1000],in[1000],temp;

    scanf("%d",&test);

    for(l=0;l<test;l++)
    {
        memset(par,0,sizeof(par));
        scanf("%d %s %s",&n,pre,in);

        for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
            {
                if(pre[i]==in[j])
                {
                    temp=par[in[j]];
                    for(k=j-1;k>-1 && par[in[k]]==temp ;k--)
                        par[in[k]]=pre[i];
                    for(k=j+1;k<n && par[in[k]]==temp ;k++)
                        par[in[k]]=pre[i];
                }
            }
        }
        post(in,pre[0],n);
    }
}

int post(char in[],char root,int n)
{

    char stack[1000],left,right,val;
    int TOP=0,i,j;

    stack[TOP++]=root;

    while(TOP)
    {
        val=stack[--TOP];
        left=right='\0';

        for(i=0;i<n;i++)
            if(par[in[i]]==val)
            {
                left=in[i];
```

```
                break;
            }
        for(i=i+1;i<n;i++)
            if(par[in[i]]==val)
            {
                right=in[i];
                break;
            }
        if(left=='\0' && right=='\0')
        {
            printf("%c",val);
            par[val]='\0';
        }
        else
        {
            stack[TOP++]=val;
            if(right!='\0')
                stack[TOP++]=right;
            if(left!='\0')
                stack[TOP++]=left;
        }
    }
    printf("\n");
    return 0;
}

/*
Input:

format: n preorder inorder
3
3 xYz Yxz
3 abc cba
6 ABCDEF CBAEDF

Output:
Yzx
cba
CBEFDA
*/
```

```
int BinarySearch(vector<int>data,int item);

int main()
{
	int i,n,temp,item,ind;
	vector<int>data;

	while(scanf("%d",&n)==1)
	{
		data.clear();
		for(i=0;i<n;i++)
		{
			scanf("%d",&temp);
			data.push_back(temp); //data should be sorted
		}
		scanf("%d",&item);
		ind=BinarySearch(data,item);
		if(ind==-1)
		{
			printf("Don't Match\n");
		}
		else
		{
			printf("index:%d\n",ind);
		}
	}
	return 0;
}

int BinarySearch(vector<int>data,int item)
{
	int low,high,mid;

	low=0;
	high=data.size();
	while(low<=high)
	{
		mid=(low+high)/2;
		if(data[mid]==item)
		{
			return mid;
		}
		else if(data[mid]<item)
		{
			low=mid+1;
		}
		else
		{
			high=mid-1;
		}
	}
	return -1;
}
```

```
#include <stdio.h>

double x[100], y[100];
int n;

int point_in_poly(double xx, double yy)
{
      int i, j, c=0;
      for (i = 0, j = n-1; i < n; j = i++)
      {
            if ( ((y[i]>yy) != (y[j]>yy)) && (xx < (x[j]-x[i]) * (yy-y[i]) /
(y[j]-y[i]) + x[i]) )
             c = !c;
      }
  return c;
}

int main()
{
      int i, check;
      double xx, yy;
      while(scanf("%d", &n)==1)
      {
            for(i=0;i<n;i++)
            {
                  scanf("%lf %lf", &x[i], &y[i]);
            }
            printf("Enter the tested point\n");

            scanf("%lf %lf", &xx, &yy);

            check = point_in_poly(xx, yy);

            if(check)
                  printf("Yes\n");
            else
                  printf("No\n");
      }
      return 0;
}
```

```
#include <stdio.h>
//int a;
double error = 0.0000000001;
double PI = 2 * acos(0);
typedef struct
{
      double x, y;
}point;
typedef struct
{
      point c; /* center of circle */
      double r; /* radius of circle */
}circle;
typedef struct
{
      double a; /* x-coefficient */
      double b; /* y-coefficient */
      double c; /* constant term */
}line;
point s; /* Superman's initial position */
point t; /* target position */

point center;
circle crcl;

//crcl.c = center;

//int ncircles; /* number of circles */
//circle c[MAXN]; /* circles data structure */
//superman()
void points_to_line(point p1, point p2, line *l)
{
      if (p1.x == p2.x)
      {
            l->a = 1;
            l->b = 0;
            l->c = -p1.x;
      }
      else
      {
            l->b = 1;
            l->a = -(p1.y-p2.y)/(p1.x-p2.x);
            l->c = -(l->a * p1.x) - (l->b * p1.y);
      }
}
double distance(point p1, point p2)
{
      return (sqrt( (p1.x-p2.x)*(p1.x-p2.x) + (p1.y-p2.y)*(p1.y-p2.y)));
}

bool parallelQ(line l1, line l2)
{
      return ( (fabs(l1.a-l2.a) <= error) && (fabs(l1.b-l2.b) <= error) );
}
```

```
bool same_lineQ(line l1, line l2)
{
      return ( parallelQ(l1,l2) && (fabs(l1.c-l2.c) <= error) );
}

point intersection_point(line l1, line l2)
{
      point p;
      if (same_lineQ(l1,l2))
      {
            printf("Warning: Identical lines, all points intersect.\n");
            p.x = p.y = 0.0;
            return p;
      }
      if (parallelQ(l1,l2) == true)
      {
            printf("Error: Distinct parallel lines do not intersect.\n");
            //return;
      }
      p.x = (l2.b*l1.c - l1.b*l2.c) / (l2.a*l1.b - l1.a*l2.b);
      if (fabs(l1.b) > error) /* test for vertical line */
            p.y = - (l1.a * (p.x) + l1.c) / l1.b;
      else
            p.y = - (l2.a * (p.x) + l2.c) / l2.b;

      return p;
}

void point_and_slope_to_line(point p, double m, line *l)
{
      l->a = -m;
      l->b = 1;
      l->c = -((l->a*p.x) + (l->b*p.y));
}

point closest_point(point p_in, line l)
{
      point p_c;

      line perp; /* perpendicular to l through (x,y) */

      if (fabs(l.b) <= error)
      {
            /* vertical line */
            p_c.x = -(l.c);
            p_c.y = p_in.y;
            return p_c;
      }
      if (fabs(l.a) <= error)
      {
            /* horizontal line */
            p_c.x = p_in.x;
            p_c.y = -(l.c);
            return p_c;
      }
      point_and_slope_to_line(p_in,1/l.a,&perp); /* normal case */
```

```
	return intersection_point(l,perp);
}

bool point_in_box(point mid,point p1,point p2)
{
	if(distance(p1, mid)+distance(p2, mid)==distance(p1, p2))
		return true;
	else
		return false;
}

int main()
{
	int test;
	//double radius;
	scanf("%d", &test);

	crcl.c.x=0.0;
	crcl.c.y=0.0;

	while(test--)
	{
		scanf("%lf %lf %lf %lf %lf", &s.x, &s.y, &t.x, &t.y, &crcl.r);

		line l; /* line from start to target position */
		point close; /* closest point */
		double d; /* distance from circle-center */
		double xray = 0.0; /* length of intersection with circles */
		double around = 0.0; /* length around circular arcs */
		double angle; /* angle subtended by arc */
		double travel; /* total travel distance */
		//int i; /* counter */
		//double asin(), sqrt();
		//double distance();

		points_to_line(s,t,&l);

		//for (i=1; i<=ncircles; i++)
		//{
		close = closest_point(crcl.c,l);
		d = distance(crcl.c,close);
		if ((d>=0) && (d < crcl.r) && point_in_box(close,s,t))
		{
			xray += 2*sqrt(crcl.r*crcl.r - d*d);
			angle = acos(d/crcl.r);
			around += ((2*angle)/(2*PI)) * (2*PI*crcl.r);
		}
		//}
		travel = distance(s, t) - xray + around;
		printf("Superman sees thru %7.3lf units, and flies %7.3lf
units\n",xray, travel);
	}
	return 0;
}
```

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <algorithm>

double pi = 3.141592653589793;
//2 * acos(0);
//3.141592653589793;
//char country[105][105];

typedef struct
{
      char name[105];
      double lat, lon ;

}cnt;

cnt country[105];

int comp(const void *a,const void *b)
{
      return (strcmp((char *)a,(char *)b));
}

int bin_s(char x[105], int left, int right)
{
      int mid;

      if(right < left)
            return -1;

      mid = floor((left+right)/2);

      if(strcmp(country[mid].name, x)==0)
            return mid;
      else if(strcmp(country[mid].name, x)>0)
            bin_s(x, left, mid-1);
      else
            bin_s(x, mid+1, right);
}

int main()
{
      char coun1[105], coun2[105];
      int i = 0, index1, index2, total ;
      double r = 6378, dis ;

      while(scanf("%s", country[i].name)==1)
      {
            if(!strcmp(country[i].name, "#"))
                  break;

            scanf("%lf %lf", &country[i].lat, &country[i].lon);
```

```
            i++;
      }

      total = i;

      qsort(country,total,sizeof(cnt),comp);

      //for(i=0;i<total;i++)
            //printf("%s %lf %lf\n", country[i].name, country[i].lat,
country[i].lon);

      while(scanf("%s%s", coun1, coun2)==2)
      {
            if((!strcmp(coun1, "#")) && (!strcmp(coun2, "#")))
                  break;

            index1 = bin_s(coun1, 0, total) ;
            index2 = bin_s(coun2, 0, total) ;

            printf("%s - %s\n", coun1, coun2);
            //printf("%d %d\n", index1, index2);

            if(index1==-1 || index2==-1)
                  printf("Unknown\n");
            else
            {
                  double dlon = country[index2].lon - country[index1].lon;
                  double dlat = country[index2].lat - country[index1].lat;
                  double a = pow((sin(dlat/2*pi/180)), 2) +
cos(country[index1].lat*pi/180) * cos(country[index2].lat*pi/180) *
pow((sin(dlon/2*pi/180)), 2);
                  double c = 2 * atan2(sqrt(a), sqrt(1-a)) ;
                  dis = r * c;

                  //dis = acos( sin(country[index1].lat * pi / 180) *
sin(country[index2].lat * pi / 180) + cos(country[index1].lat * pi / 180) *
cos(country[index2].lat * pi / 180) * cos((country[index1].lon -
country[index2].lon) *pi /180) ) * r;

                  printf("%.0lf km\n", dis);
            }
      }

      return 0;
}
```

```
#include<stdio.h>
#include<math.h>
#include<algorithm>

using namespace std;


typedef struct
{
    double x, y;
}point;

point pnts[102],stk[102],piv;

int cross(point a,point b,point c){
      if((b.x-a.x)*(c.y-a.y) > (c.x-a.x)*(b.y-a.y))
            return 1;
      else if((b.x-a.x)*(c.y-a.y) < (c.x-a.x)*(b.y-a.y))
            return -1;
      return 0;
}

int comp(const void *a, const void *b)
{
      int cr;
      point *c = (point*) a;
      point *d = (point*) b;

      cr = cross(piv,*d,*c);
      if(cr)
            return cr;
      if((pow( (piv.x - c->x), 2) + pow( (piv.y - c->y), 2) ) > ( pow( (piv.x
- d->x),2) + pow( (piv.y - d->y), 2) ) )
            return 1;
      return -1;
}

int main()
{
    int total, i;

    scanf("%d", &total);

      scanf("%lf%lf", &pnts[0].x, &pnts[0].y);

      piv = pnts[0];

    for(i=1;i<total;i++)
    {
        scanf("%lf%lf", &pnts[i].x, &pnts[i].y);

            if(pnts[i].y < piv.y)
                  piv = pnts[i];
            else if(pnts[i].y == piv.y)
```

```
            {
                  if(pnts[i].x < piv.x)
                        piv = pnts[i];
            }
      }

    qsort(pnts, total, sizeof(point), comp);

    int top = 0;

      stk[top++] = pnts[0];
      stk[top++] = pnts[1];
      stk[top++] = pnts[2];


    for(i=3;i<total;i++)
    {

        while(cross(stk[top-2],stk[top-1],pnts[i]) < 0)
                  top--;

        stk[top++] = pnts[i];
    }

    for(i=0;i<top;i++)
    {
        printf("%lf %lf\n", stk[i].x, stk[i].y);
    }


    return 0;
}
```

```
#include <iostream>
#include <string>
#include <vector>
#include <sstream>
#include <algorithm>
#include <stack>
#include <queue>
#include <set>
#include <cmath>
#include <map>

using namespace std;

#define PI 2*acos(0)

int main()
{
    int i,len;

    //Vector Implements
    vector<int>a;

    a.clear();
    for(i=0;i<5;i++)
        a.push_back(i+5);

    len=a.size();

    for(i=0;i<len;i++)
        printf("%d\n",a[i]);

    a.erase(a.begin()+3);

    for(i=0;i<a.size();i++)
        printf("%d\n",a[i]);



    //string Implement
    string str="We think in generalities, but we live in details.";
    string str2;
    string::iterator it;

    str2 = str.substr (12,12);
    cout<<str2<<endl;

    if(str.find(str2)!=-1)
        cout<<"Found Data\n";

    reverse(str.begin(),str.end());

    cout<<str<<endl;
```

```
//String Conversion
istringstream iss;
iss.str(str);
while(iss>>str2)
{
    cout<<str2<<endl;
}

str="1245 56898 5689 1245 124545";
iss.clear();
iss.str(str);
while(iss>>i)
{
    cout<<i<<endl;
}

for(it=str.begin();it!=str.end();it++)
    cout<<*it;
cout<<endl;


//Map Opertion
map<char,int> mymap;
map<char,int>::iterator x;

mymap['b'] = 100;
mymap['a'] = 200;
mymap['c'] = 300;
for ( x=mymap.begin() ; x != mymap.end(); x++ )
    cout << (*x).first << " => " << (*x).second << endl;

mymap.clear();

mymap['c']=50;
mymap['b']=100;
mymap['a']=150;
mymap['d']=200;
x=mymap.find('b');
mymap.erase (x);
mymap.erase (mymap.find('d'));
for ( x=mymap.begin() ; x != mymap.end(); x++ )
    cout << (*x).first << " => " << (*x).second << endl;

cout << "mymap.size() is " << (int) mymap.size() << endl;

//Queue Operation
queue<int> myqueue;
int myint;

cout << "Please enter some integers (enter 0 to end):\n";

for (int i=0; i<5; ++i) myqueue.push(i+5);

cout << "myqueue contains: ";
while (!myqueue.empty())
{
    cout << " " << myqueue.front();
```

```
    myqueue.pop();
}

//Stack Operation
stack<int> mystack;

for (int i=0; i<5; ++i) mystack.push(i);

cout << "\nPopping out elements...";
while (!mystack.empty())
{
    cout << " " << mystack.top();
    mystack.pop();
}
cout << endl;

//Priority Queue
priority_queue<int> mypq;

mypq.push(30);
mypq.push(100);
mypq.push(25);
mypq.push(40);

cout << "Popping out elements...";
while (!mypq.empty())
{
    cout << " " << mypq.top();
    mypq.pop();
}
cout << endl;


//Set
set<int> myset;
set<int>::iterator y;

for (int i=1; i<=5; i++) myset.insert(i*10);
cout << "myset contains:";
for (y=myset.begin(); y!=myset.end(); y++)
    cout << " " << *y;
cout << endl;
y=myset.find(20);
myset.erase (y);
myset.erase (myset.find(40));

cout << "myset contains:";
for (y=myset.begin(); y!=myset.end(); y++)
    cout << " " << *y;
cout << endl;
cout<<"size:"<<(int)myset.size()<<endl;
myset.clear();
cout<<"size:"<<(int)myset.size()<<endl;


//Algorithm STL
 int myints[] = {1,2,3,4,5,4,3,2,1};
```

```
    vector<int> v(myints,myints+9);
    vector<int>::iterator m;
    sort (v.begin(), v.end());

    cout << "looking for a 3... ";
    if (binary_search (v.begin(), v.end(), 3))
        cout << "found!\n"; else cout << "not found.\n";
    m=find(v.begin(),v.end(),3);
    cout<<"value:"<<*m<<endl;
    cout << "Min:" << *min_element(v.begin(),v.end())<<endl;
    cout << "Max:" << *max_element(v.begin(),v.end())<<endl;

    //Next Permutation
    int b[] = {1,2,3};

    cout << "The 3! possible permutations with 3 elements:\n";

    sort (b,b+3);

    do {
        cout << b[0] << " " << b[1] << " " << b[2] << endl;
    } while ( next_permutation (b,b+3) );

    //Prev Permutation
    printf("\n\n\n");
    sort (b,b+3);
    reverse (b,b+3);

    do {
        cout << b[0] << " " << b[1] << " " << b[2] << endl;
    } while ( prev_permutation (b,b+3) );


    //Cmath
    //tan2
    double p, q, result;
    p = -10.0;
    q = 10.0;
    result = atan2(p,q) * 180 / PI;
    printf ("arc tan(x=%lf, y=%lf) is %lf deg\n",p,q,result);

    //exponential
    double param;
    param = 5.0;
    result = exp (param);
    printf ("The exponential value of %lf is %lf.\n", param, result );
    return 0;
}
```

```
#include<iostream>

using namespace std;

struct node
{
    node *left,*right,*par;
    int item;
} *root,*tree,*stack[100];
typedef struct node node;

void construct_tree();
void print_tree();
void insert(int item);
void Delete(int item);
node trans_node(node *temp);
node Tree_Successor(node *x);
node Tree_Minimum(node *x);
int main()
{
    construct_tree();
    print_tree();
    //insert(12);
    //print_tree();
    Delete(15);
    print_tree();
    return 9;
}
void construct_tree()
{
    int a[]={15,5,16,3,12,20,10,13,18,23,6,7},i;

    //here root data is 15

    root=tree=new node();
    root->par=new node();

    for(i=0;i<12;i++)
    {
        tree=root;
        while(1)
        {
            if(tree->item==0)//check for null value
            {
                tree->item=a[i];
                tree->left=new node();
                tree->left->par=tree;
                tree->right=new node();
                tree->right->par=tree;
                break;
            }
            else if(a[i]<tree->item)
                tree=tree->left;
            else tree=tree->right;
```

```
        }
    }
}
void print_tree()
{
    int TOP;

    tree=root;
    TOP=0;

    while(TOP!=-1)
    {
        if(tree->item==0)
            tree=stack[--TOP];
        else
        {
            printf("%d\n",tree->item);
            stack[TOP++]=tree->right;
            tree=tree->left;
        }
    }
    printf("\n\n");
}
void insert(int data)
{
    tree=root;

    while(tree->item)  //check for null
    {
        if(data>tree->item)
            tree=tree->right;
        else tree=tree->left;
    }
    tree->item=data;
    tree->left=new node();
    tree->left->par=tree;
    tree->right=new node();
    tree->right->par=tree;
}
void Delete(int data)
{
    node *y,*z,*x;

    tree=root;

    while(tree->item && tree->item!=data)
    {
        if(data>tree->item)
            tree=tree->right;
        else tree=tree->left;
    }
    if(tree->item!=data)
    {
        printf("Not Found This Data!!!\n\n");
        return ;
    }
    z=tree;
```

```
    if((z->left->item==0) || (z->right->item==0))
        y=z;
    else
        *y=Tree_Successor(z);
    if(y->left->item!=0)
        x=y->left;
    else
        x=y->right;

    if(x->item!=0)
        x->par=y->par;
    if(y->par->item==0)
        root=x;
    else if(y==y->par->left)
            y->par->left=x;
        else y->par->right=x;
    if(y!=z)
        z->item=y->item;
    return ;


}

node trans_node(node *temp)
{
    printf("%d\n",temp->item);
    return *temp;
}

node Tree_Successor(node *x)
{
    node *y;

    if(x->right->item!=0)
        return Tree_Minimum(x->right);
    y=x->par;
    while(y->item!=0 && x==y->right)
    {
        x=y;
        y=y->par;
    }
    return *y;
}

node Tree_Minimum(node *x)
{
    while(x->left->item!=0)
        x=x->left;
    return *x;
}
```