Contents

0.1 Graph

0.1.1 Euler Tour

```
#include<stdio.h>
const int MAX=1025;
int v[MAX],link[MAX][MAX],c[MAX],p,i;
void go(int s)
{
    while(v[s])
    {
        for(i=1;i<MAX && !link[i][s];i++);
        v[i]--,v[s]--;
        link[i][s]--;link[s][i]--;
        go(i);
    }
    c[p++]=s;
}
int main(){
    int s,f,l1,l2;
    while(1){
        for(s=0;s<MAX;s++)
        {
            v[s]=link[s][s]=0;
            for(f=0;f<s;f++)
                link[s][f]=link[f][s]=0;
        }
        scanf("%d",&f);
        while(f--)
        {
            scanf("%d %d",&l1,&l2);
            v[l1]++,v[l2]++;
            link[l1][l2]++,link[l2][l1]++;
        }
        for(s=1;s<MAX && v[s]%2!=1;s++);
        if(s>=MAX)for(s=1;v[s]==0;s++);
        p=0;
        go(s);
        while(p--)printf("%d\n",c[p]);
    }
    return 0;
}
```

0.1.2 Minimum Spaning tree and Disjoint sets

```
#include<algorithm>
using namespace std;
const int V=1000;
int n,m,p[V],d[V],c[V][V];
int find(int v){
    if(p[v]==v)return v;
    return p[v]=find(p[v]);
}
void uni(int a,int b){
    a=find(a),b=find(b);
    if(d[a]<d[b])p[a]=b;
    else p[b]=a;
    if(d[a]==d[b])d[a]++;
}
void init(int n){
    while(n--)p[n]=n,d[n]=0;
}
struct pt{
    int x,y;
    int operator+(pt a){return (x-a.x)*(x-a.x)+(y-a.y)*(y-a.y);}
}pos[V];
struct ed{
    int s,t,c;
    bool operator<(ed x)const{return c<x.c;}
}s[V*V];
int kuskal(){
    std::sort(s,s+m);
    init(n);
    int i,sum;
    for(i=0;i<m;i++){
        if(find(s[i].s)==find(s[i].t))continue;
        uni(s[i].s,s[i].t);
        sum+=s[i].c;
    }
    return sum;
}
int prim(){
    const int inf=2147483647;
    int i,j,sum=0;
    for(i=0;i<n;i++)
        if(c[0][i]<inf)
            d[i]=c[0][i],p[i]=0;
        else d[i]=inf,p[i]=i;
    p[0]=-1;
    while(1){
        for(i=0;i<n;i++)
            if(d[i]>=0 && d[i]<inf)break;
        if(i>=n)break;
        for(j=i+1;j<n;j++)
            if(d[j]>=0 && d[j]<d[i])i=j;
        for(j=0;j<n;j++)
            if(c[i][j]<d[j])
                d[j]=c[i][j],p[j]=i;
        sum+=d[i];
        d[i]=-1;
    }
    return sum;
}
```

0.1.3 Directed MST

```
const int V=1001;
struct E{short s,t,c;}c[50000];
int n,to[V],help[V],back[V],pre[V];
char u[50000];
int d[V],p[V],p0[V];
int find(int v)
{
    if(p[v]==v)return v;
    return(p[v]=find(p[v]));
}
void uni(int a,int b)
{
    a=find(a),b=find(b);
    if(d[a]<d[b])p[a]=b;
    else p[b]=a;
    if(d[a]==d[b])d[a]++;
}
void clear(){for(int i=0;i<n;i++)p[i]=i,d[i]=0;}
void backup(){for(int i=0;i<n;i++)p0[i]=p[i];}
void recover(){for(int i=0;i<n;i++)p[i]=p0[i];}
main()
{
    int i,j,k,m,sum,T,C=1;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d %d",&n,&m);
        clear();
        for(i=0;i<m;i++)
        {
            scanf("%d %d %d",&c[i].s,&c[i].t,&c[i].c),u[i]=1;
            if(c[i].s==c[i].t || c[i].t==0)i--,m--;
        }//Discard the edges entering the root or itself
        printf("Case #%d: ",C++);
        if(m==0)
        {
            if(n>1)puts("Possums!");
            else puts("0");
            continue;
        }
        for(j=k=sum=0;m>0;m=j,j=1)
        {   //Select min-cost entering edge for every node
            for(i=0;i<n;i++)to[i]=-1,help[i]=0;
            for(i=0;i<m;i++)
               if(to[c[i].t]<0 || c[i].c<c[to[c[i].t]].c)
                    to[c[i].t]=i;
            for(i=0;k>=0 && i<n;i++)//Check if every node have entering node
                if(i!=k && p[i]==i && to[i]<0)k=-1;
            if(k<0)break;
            backup();
            for(i=j=0;i<n;i++)
                if(to[i]>=0)
                {
                    if(find(c[to[i]].s)==find(c[to[i]].t))
                        help[i]=1,j++; //Mark cycles
                    else uni(c[to[i]].s,c[to[i]].t);
                    sum+=(back[c[to[i]].t]=c[to[i]].c);
                    pre[c[to[i]].t]=c[to[i]].s;
                }
            if(!j)break;
            recover();
            for(i=0;i<n;i++)
                if(help[i]) //Merging cycles
                    while(find(i)!=find(pre[i]))
                        uni(i,pre[i]),i=pre[i];
            k=find(k);
            for(i=0;i<m;i++)
                if(u[i])
                {   //Reassign edge weight
                    c[i].c-=back[c[i].t];
                    c[i].s=find(c[i].s),c[i].t=find(c[i].t);
                    if(c[i].s==c[i].t)u[i]=0;
                }   //Discard inner edges
            for(i=j=0;i<m;i++)
                if(u[i])c[j]=c[i],u[j++]=1;
        }
        if(k<0 || j)puts("Possums!");
        else printf("%d\n",sum);
    }
}
```

0.1.4 Matching

```
const int V=1005;
char p[V],c[V][V];
int t;
char go(int v){
    if(p[v])return 0;
    if(v==t)return 1;
    p[v]=1;
    for(int i=1;i<=t;i++)
        if(c[v][i]-- && go(i))
            return ++c[i][v];
        else c[v][i]++;
    return 0;
}
int flow(){
    int i,j=0;
    for(i=0;i<=t;i++)p[i]=0;
    while(go(0))
        for(j++,i=0;i<=t;i++)p[i]=0;
    return j;
}
```

0.1.5 Matching(Hopcroft-Karp)

```
const int V=1024,inf=2147483647;    //low flow
int t,c[V][V],p[V],d[V],u[V],q[V];
int sp(){
    int i,j,k;
    for(i=0;i<=t;i++)
        d[i]=inf;
    d[q[0]=t]=0;
    for(i=0,j=1;d[0]>=inf && i<j;i++)
        for(k=0;k<=t;k++)
            if(c[k][q[i]] && d[k]>=inf)
                d[q[j++]=k]=d[q[i]]+1;
    return d[0];
}
int go(int v){
    if(v==t)return 1;
    int sum=0;
    for(;u[v]<=t;u[v]++)
        if(d[v]==d[u[v]]+1){
            if(c[v][u[v]]-- && go(u[v])){
                c[u[v]][v]++;
                if(v)return 1;
                sum++;
            }
            else c[v][u[v]]++;
        }
    return sum;
}
int flow(){
    int i,sum=0;
    while(sp()<inf){
        for(i=0;i<=t;i++)u[i]=0;
        sum+=go(0);
    }
    return sum;
}
```

0.1.6 Max-Flow

```
const int MaxV=200;
struct list{
    int v;
    list *prev,*next;
}*head;
int t,h[MaxV],e[MaxV],n[MaxV],c[MaxV][MaxV];
inline int min(int a,int b){
    if(a<b)return a;
    return b;
}
void makelist(){
    int i;
    list *ptr;
    head=new list;
    head->prev=head->next=NULL;
    ptr=head;
    for(i=1;i<t-1;i++,ptr=ptr->next){
        ptr->v=i;
        ptr->next=new list;
        ptr->next->prev=ptr;
    }
    ptr->v=t-1;
    ptr->next=NULL;
}
void cleanlist(list *&ptr=head){
    if(ptr){
        cleanlist(ptr->next);
        delete ptr;
        ptr=NULL;
    }
}
void tofront(list *v){
    v->prev->next=v->next;
    if(v->next!=NULL)
        v->next->prev=v->prev;
    v->next=head;
    head->prev=v;
    v->prev=NULL;
    head=v;
}
void push(int v,int u){
    int f=min(c[v][u],e[v]);
    e[v]-=f;
    e[u]+=f;
    c[v][u]-=f;
    c[u][v]+=f;
}
void relabel(int v){
    int i,min;
    for(i=0,min=2*t;i<=t;i++)
        if(c[v][i]>0 && h[i]+1<min)
            min=h[i]+1;
    h[v]=min;
}
void discharge(int v){
    while(e[v]>0){
        if(n[v]>t)
            relabel(v),n[v]=0;
        else if(c[v][n[v]]>0 && h[v]==h[n[v]]+1)
            push(v,n[v]);
        else n[v]++;
    }
}
int flow(){
    int i,v,lh;
    list *ptr;
    for(i=0;i<=t;i++)h[i]=n[i]=e[i]=0;
    for(i=0;i<=t;i++){
        if(c[0][i]==0)
            continue;
        e[i]=c[0][i];
        c[0][i]-=e[i];
        c[i][0]+=e[i];
    }
    h[0]=t+1;
    makelist();
    for(ptr=head;ptr!=NULL;ptr=ptr->next){
        v=ptr->v;
        lh=h[v];
        discharge(v);
        if(h[v]>lh && head!=ptr)
            tofront(ptr);
    }
    return e[t];
    cleanlist();
}
```

0.1.7 Min-cost Matching

```
const long long inf=9223372036854775807ll;
const int V=100;
char u[V];
int n,p[V],q[V*V],cap[V][V],cost[V][V];
long long d[V+1];
long long sp(){
    int i,j,k;
    for(i=0;i<n;i++)d[i]=inf,u[i]=1;
    q[0]=d[0]=u[0]=0;
    for(i=0,j=1;i<j;i++){
        u[q[i]]=1;
        for(k=0;k<n;k++){ //Change here if directed
            if(cap[q[i]][k]==2 && d[q[i]]-cost[k][q[i]]<d[k]){
                d[k]=d[p[k]=q[i]]-cost[k][q[i]];
                if(u[k])u[q[j++]=k]=0;
            }
            if(cap[q[i]][k]==1 && d[q[i]]+cost[q[i]][k]<d[k]){
                d[k]=d[p[k]=q[i]]+cost[q[i]][k];
                if(u[k])u[q[j++]=k]=0;
            }
        }
    }
    if(d[n-1]<inf)return d[n-1];
    return -1;
}
long long flow(int m){
    long long tmp,sum=0;
    while(m>0 && (tmp=sp())>=0){
        sum+=tmp;
        m--;
        for(int i=n-1;i>0;i=p[i])
            cap[p[i]][i]--,cap[i][p[i]]++;
    }
    return sum;
}
```

0.1.8 Min-cost max flow

```
#include<stdio.h>
#define V 100
#define BIG 99999999
#define S 0 //source
#define T 1 //sink

int map[V][V],cost[V][V];  //when not min-cost, set ALL cost[i][j] to 1

struct dp{
    int cost;
    int pre;
};

dp DP[V][V];

int flag;
int min_cost_max_flow(){  //return cost
    flag=0;
    int i,j,k,sum=0,q[V],ptr;
    for(i=0;i<V;i++)
        DP[0][i].cost=BIG;
    DP[0][S].cost=0;
    ptr=1;
    q[0]=S;
    for(i=1;i<=V;i++){
        for(j=0;j<V;j++){
            DP[i][j].cost=DP[i-1][j].cost;
            DP[i][j].pre=j;
            for(k=0;k<ptr;k++){
                if(map[q[k]][j]>0 &&
                   DP[i-1][q[k]].cost+cost[q[k]][j]<DP[i][j].cost){
                    DP[i][j].cost=DP[i-1][q[k]].cost+cost[q[k]][j];
                    DP[i][j].pre=q[k];
                }
            }
        }
        ptr=0;
        for(j=0;j<V;j++){
            if(DP[i][j].cost<DP[i-1][j].cost)
                q[ptr++]=j;
        }
    }
    if(DP[V][T].cost==BIG){
        flag=1;
        return 0;
    }
    int min=BIG,tmp;
    tmp=T;
    for(i=V;i>0;i--){
        if(tmp==DP[i][tmp].pre)
            continue;
        if(min>map[DP[i][tmp].pre][tmp])
            min=map[DP[i][tmp].pre][tmp];
        tmp=DP[i][tmp].pre;
    }
    tmp=T;
    for(i=V;i>0;i--){
        if(tmp==DP[i][tmp].pre)
            continue;
        map[DP[i][tmp].pre][tmp]-=min;
        map[tmp][DP[i][tmp].pre]+=min;
        sum+=cost[DP[i][tmp].pre][tmp]*min;
        tmp=DP[i][tmp].pre;
    }
    return sum;
}
```

0.2 Search

0.2.1 15-Puzzle(IDA*)

```
#include<cstdio>
#include<cstdlib>
#define vaild(x,y) (x>=0 && x<n && y>=0 && y<n)
const int n=4,dx[]={1,0,-1,0},dy[]={0,1,0,-1},rev[]={2,3,0,1};
char path[1000],dirt[]="DRUL";
int x,y,step,score,next,bound,s[n][n],num[n*n][n][n];
bool dfs(){
    if(step+score>bound)    {
        if(step+score<next)
            next=step+score;
        return false;
    }
    if(!score)    {
        return true;
        path[step]=0;
    }
    for(int i=0;i<4;i++)
        if(path[step-1]!=dirt[rev[i]] && vaild(x+dx[i],y+dy[i]))        {
            score-=num[s[x][y]][x][y];
            s[x+dx[i]][y+dy[i]]=s[x][y];
            s[x+=dx[i]][y+=dy[i]]=0;
            score+=num[s[x][y]][x][y];
            path[step++]=dirt[i];
            if(dfs())
                return true;
            step--;
            score-=num[s[x][y]][x][y];
            s[x-dx[i]][y-dy[i]]=s[x][y];
            s[x-=dx[i]][y-=dy[i]]=0;
            score+=num[s[x][y]][x][y];
        }
    return false;
}
main(){
    int i,j,k,l,t;
    scanf("%d",&t);
    for(k=n*n-2;k>=0;k--)
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                num[k+1][i][j]=abs(k/n-i)+abs(k%n-j);
    while(t--){
        for(i=l=score=0;i<n;i++)
            for(j=0;j<n;j++){
                scanf("%d",&s[i][j]);
                if(s[i][j]==0){
                    x=i,y=j;
                    continue;
                }
                score+=num[s[i][j]][i][j];
                for(k=i*n+j;k>=0;k--)
                    if(s[k/n][k%n]>s[i][j])l++;
            }
        if((l+x+1)&1){
            puts("This puzzle is not solvable.");
            continue;
        }
        step=1,bound=score;
        while(1){
            next=1000;
            if(dfs())break;
            bound=next;
        }
        puts(path);
    }
}
```

0.2.2 Sticks

```
#include<stdio.h>
int y,n,m,min,max,use[51],num[51],able[10000];
void dfs(int len,int d,int i){
   if(y)return;
   if(!d){
       y=1;
       return;
   }
   if(!len){
       while(!use[i])i--;
       use[i]--;
       if(i==m)
           dfs(0,d-1,max);
       else if(i+min<=m)
           dfs(i,d,i);
       use[i]++;
       return;
   }
   for(;i>0 && !y;i--)
       if(use[i] && i+len<=m){
           use[i]--;
           if(len+i==m){
               dfs(0,d-1,max);
               use[i]++;
               return;
           }
           else if(i+min<=m)
               dfs(len+i,d,i);
           use[i]++;
       }
}
main(){
int i,k,sum,stick;
while(scanf("%d",&n) && n){
   for(min=51,max=i=sum=0;i<n;i++){
       scanf("%d",&stick);
       sum+=stick;
```

```
        num[stick]++;
        max>?=stick;
        min<?=stick;
    }
    for(y=0,i=max;!y && i<=50;i++)
        if(sum%i==0){
            m=i;
            for(k=1;k<=50;k++)use[k]=num[k];
            dfs(0,sum/i,max);
        }
    printf("%d\n",y?m:sum);
}
}
```

0.3 Mathematics

0.3.1 GCD

```
#include <iostream>

using namespace std;

int gcd(int f, int g, int &a, int &b){
    int d;
    if(!g){
        a=1;
        b=0;
        return f;
    }
    d=gcd(g,f%g,b,a);
    b=b-a*(f/g);
    return d;
}

int main(){
    int a,b,d,x,y;
    cin >> a >> b;
    d=gcd(a,b,x,y);
    cout << "gcd(" << a << "," << b << ")=" << d << ", "
        << a << "*" << x << "+" << b << "*" << y << "=" << d << endl;
    return 0;
}
```

0.3.2 Factorization and Primality Test

```
#include<cmath>
#include<cstdio>
#include<algorithm>
using namespace std;
long long Rand(){
    return rand()*(1ll<<48)+rand()*(1ll<<32)+rand()*(1ll<<16)+rand();
}
long long mul(long long a,long long b,long long m){
    long long i,res=0;
    for(i=1;i<=b;i*=2,(a*=2)%=m)
        if(b&i)(res+=a)%=m;
    return res;
}
long long pow(long long n,long long k,long long m){
    if(k==0)return 1;
    if(k%2==1)
        return mul(n,pow(n,k-1,m),m);
    n=pow(n,k/2,m);
    return mul(n,n,m);
}
bool witness(long long a,long long n){
    long long x,y,u,t;
    for(u=n-1,t=0;u%2==0;u/=2,t++);
    x=pow(a,u,n);
    while(t--){
        y=x;
        x=pow(x,2,n);
        if(x==1 && y!=1 && y!=n-1)
            return 1;
    }
    return x!=1;
}
bool mr(long long n,int s=25){
    if(n-1>=2 && witness(2,n))return 0;
    if(n-1>=3 && witness(3,n))return 0;
    if(n-1>=7 && witness(7,n))return 0;
    if(n-1>=61 && witness(61,n))return 0;
    if(n-1>=24251 && witness(24251,n))return 0;
    if(n==46856248255981ll)return 0;
    return 1;
}
long long gcd(long long a,long long b){
    while((a%=b)&&(b%=a));
    return a+b;
}
namespace g{long long abs(long long x){return x<0?-x:x;}}
int _c=1;
long long _f(long long x,long long n)
{return(mul(x,x,n)+_c)%n;}
long long go(long long n){
    long long x,y,d=1;
        x=y=Rand()%n;
        while(d==1){
            x=_f(x,n);
            y=_f(_f(y,n),n);
            d=gcd(g::abs(y-x),n);
        }
        if(d!=n)return d;
    return d;
}
void fa(long long n,int& fn,long long s[]){
    long long x;
    while(n>1 && n%2==0)
        s[fn++]=2,n/=2;
    while(!mr(n)){
        for(_c=1,x=n;x==n;_c=1+Rand()%(n-1))x=go(n);
        if(x<0)break;
        n/=x;
        fa(x,fn,s);
    }
    if(n>1)s[fn++]=n;
}
main(){
    int i,j,m;
    long long n,k,s[70];
    while(scanf("%I64d",&n)==1){
        m=0;
        while(n%2==0)n>>=1;
        fa(n,m,s);
        std::sort(s,s+m);
        for(i=0,k=1;i<m;i+=j){
            for(j=0;i+j<m && s[i+j]==s[i];j++);
            k*=j+1;
        }
        printf("%I64d\n",k);
    }
}
```

0.3.3 Linear Congruences

```
bool linearCongruences(const vector<Int> &a,
                       const vector<Int> &b,
                       const vector<Int> &m,
                       Int &x, Int &M) {
  int n = a.size();
  x = 0; M = 1;
  REP(i, n) {
    Int a_ = a[i] % M, b_ = b[i] - a[i] * x, m_ = m[i];
    Int y, t, g = extgcd(a_, m_, y, t);
    if (b_ % g) return false;
    b_ /= g; m_ /= g;
    x += M * (y * b_ % m_);
    M *= m_;
  }
  x = (x + M) % M;
  return true;
}
```

0.3.4 Number-Theoretic Transform

```
#include<cstdio>
#include<algorithm>
using namespace std;
inline int pow(long long n,int k,int m){
    unsigned i;
    int a;
    for(a=i=1;i<=k;i*=2,n=(n*n)%m)
        if(k&i)a=(a*n)%m;
    return a;
}
inline int rev(int n,int k){
    int i=0;
    while(k--)(i<<=1)+=n&1,n/=2;
    return i;
}
const long long p=1107296257,r=10;
void ntt(bool f,int& n,int s[]){
    int i,j,k;
    long long x,w;
    for(j=1,k=0;j<n ;j<<=1,k++);
    for(;n<j;n++)s[n]=0;
    for(i=0;i<n;i++)
        if(i<rev(i,k))
            swap(s[i],s[rev(i,k)]);
    for(i=2;i<=n;i*=2){
        w=pow(pow(r,(p-1)/n,p),f?p-1-n/i:n/i,p);
        for(j=0;j<n;j+=i)
```

```
            for(k=0,x=1;k<i/2;k++){
                int &a=s[j+k],&b=s[j+k+i/2];
                b=(b*x)%p;
                a=(a*1ll+b)%p;
                b=(a+p+p-b-b)%p;
                x=x*w%p;
            }
    }
    x=pow(n,p-2,p);
    if(f)for(j=0;j<n;j++)
        s[j]=(s[j]*x)%p;
}
void mul(int n,int a[],int b[]){
    for(int i=0;i<n;i++)
        a[i]=(1ll*a[i]*b[i])%p;
}
void print(int n,int s[]){
    for(int i=0;i<n;i++)
        printf("%d ",s[i]);
    puts("");
}
int s[1000000],t[1000000];
char a[250000],b[250000];
main(){
    int i,j,n1,n2;
    while(scanf("%s %s",a,b)==2){
        for(n1=0;a[n1];n1++);
        for(n2=0;b[n2];n2++);
        for(i=0;n1--;i++)
            s[i]=a[n1]-'0';
        n1=i;
        for(i=0;n2--;i++)
            t[i]=b[n2]-'0';
        n2=i;
        for(i=2;i<n1+n2 || i<(n1>?n2)*2;i*=2);
        for(;n1<i;n1++)s[n1]=0;
        for(;n2<i;n2++)t[n2]=0;
        ntt(0,n1,s);
        ntt(0,n2,t);
        mul(n2,t,s);
        ntt(1,n2,t);
        t[n2]=0;
        for(i=0;i<n2;i++){
            t[i+1]+=t[i]/10;
            t[i]%=10;
        }
        if(t[n2])n2++;
        while(n2 && !t[n2-1])n2--;
        printf("%d",t[n2-1]);
        for(i=n2-2;i>=0;i--)
            printf("%d",t[i]);
        puts("");
    }
}
```

0.3.5 Permutation

```
#include<algorithm>
long long f[50];
long long go(int num[],int n,int t){
    int i,j,k;
    long long s1[25],s2[25];
    if(n<=1)return 1;
    for(j=0;j<=t;j++)s1[j]=0;
    for(j=0;j<=num[0] && j<=t;j++)
        s1[j]=f[t]/f[j];
    for(i=1;i<n;i++){
        for(j=0;j<=t;j++)s2[j]=0;
        for(j=0;j<=num[i] && j<=t;j++)
            for(k=0;j+k<=t;k++)
                s2[j+k]+=s1[k]/f[j];
        for(j=0;j<=t;j++)s1[j]=s2[j];
    }
    return s1[t];
}
int count(int s[],int n,int num[]){
    int i,j;
    std::sort(s,s+n);
    for(i=j=0;i<n;j++)
    {
        s[j]=s[i];
        for(num[j]=0;i<n && s[i]==s[j];i++)num[j]++;
    }
    return j;
}
void make(int s1[],int u1[],const int& n1,int s2[],int n2,long long k){
    int i,j;
    long long l;
    for(i=0;i<n2;i++)
        for(j=0;j<n1;j++)
        {
            if(!u1[j])continue;
            s2[i]=s1[j],u1[j]--;
            l=go(u1,n1,n2-i-1);
            if(k<=l)break;
            k-=l,u1[j]++;
        }
}
long long get(int s1[],int u1[],int n1,int s2[],int n2){
    int i,j;
    long long k=0;
    for(i=0;i<n2;i++)
        for(j=0;j<n1;j++)
        {
            u1[j]--;
            if(s2[i]==s1[j])break;
            k+=go(u1,n1,n2-i-1);
            u1[j]++;
        }
    return k;
}
```

0.3.6 Order Sum

```
#include<stdio.h>
//O(nlogn),
int temp[100];  //ed-st<100
int ordersum(int st, int ed, int* arr){
    if(ed-st<=1) return 0;
    int mid=(st+ed)/2, len1=mid-st, len2=ed-mid;
    int num,a,b,c=0,n1,n2;
    num=ordersum(st,mid,arr)+ordersum(mid,ed,arr);
    for(a=st,b=mid;a<mid && b<ed;){
        if(arr[a]<arr[b]){
            temp[c++]=arr[a];
            num+=(ed-b);
            a++;n1++;
        }
        else{
            temp[c++]=arr[b];
            //num+=(mid-a)
            b++;n2++;
        }
    }
    if(a<mid)
        for(;a<mid;a++)temp[c++]=arr[a];
    else
        for(;b<ed;b++)temp[c++]=arr[b];
    for(a=0;a<c;a++)
        arr[st+a]=temp[a];
    return num;
}
```

0.3.7 Finding roots(formula)

```
/* $Id: root_of_equation.c,v 1.1 2002/02/25 09:46:50 kcwu Exp $ */
#include <math.h>
#define EQN_EPS 1e-9

static int IsZero(double x)
{
    return x < - EQN_EPS && x < EQN_EPS;
}
int SolveQuadric(double c[ 3 ], double s[ 2 ])
{
    double p, q, D;
    /* x^2 + px + q = 0 */
    p = c[1] / (2 * c[2]);
    q = c[0] / c[ 2 ];
    D = p * p - q;

    if (IsZero(D)) {
        s[ 0 ] = - p;
        return 1;
    } else if (D < 0) {
        return 0;
    } else if (D > 0) {
        double sqrt_D = sqrt(D);
        s[ 0 ] =   sqrt_D - p;
        s[ 1 ] = - sqrt_D - p;
        return 2;
    }
}
int SolveCubic(double c[ 4 ], double s[ 3 ])
{
    int    i, num;
    double sub;
    double A, B, C;
    double sq_A, p, q;
    double cb_p, D;

    /* x^3 + Ax^2 + Bx + C = 0 */
    A = c[ 2 ] / c[ 3 ];
    B = c[ 1 ] / c[ 3 ];
    C = c[ 0 ] / c[ 3 ];
    sq_A = A * A;
    p = 1.0/3 * (-1.0/3 * sq_A + B);
    q = 1.0/2 * (2.0/27* A * sq_A - 1.0/3 * A * B + C);

    /* Cardano  */
    cb_p = p * p * p;
    D = q * q + cb_p;
    if(IsZero(D)) {
        if(IsZero(q)) {
            s[0] = 0;
            num = 1;
        } else {
            double u=cbrt(-q);
            s[0]=2*u;
            s[1]=-u;
            num=2;
        }
    } else if(D<0) {
        double phi=1.0/3*acos(-q/sqrt(-cb_p));
        double t=2*sqrt(-p);
        s[0]=t*cos(phi);
        s[1]=-t*cos(phi+M_PI/3);
        s[2]=-t*cos(phi-M_PI/3);
        num=3;
    } else {
        double sqrt_D=sqrt(D);
        double u=cbrt(sqrt_D-q);
        double v=-cbrt(sqrt_D+q);
        s[0]=u+v;
        num=1;
    }
    sub=1.0/3*A;
    for(i=0;i<num;i++)
        s[i]-=sub;
    rutern num;
}
int SolveQuartic(double c[5], double s[4])
{
    double coeffs[4];
    double z, u, v, sub;
    double A, B, C, D;
    double sq_A, p, q, r;
    int    i, num;

    /* x^4+Ax^3+Bx^2+Cx+D=0 */
    A=c[3]/c[4];
    B=c[2]/c[4];
    C=c[1]/c[4];
    D=c[0]/c[4];
    sq_A=A*A;
    p=-3.0/8*sq_A+B;
    q=1.0/8*sq_A*A-1.0/2*A*B+C;
    r=-3.0/256*sq_A*sq_A+1.0/16*sq_A*B-1.0/4*A*C+D;

    if(IsZero(r)) {
        /* y(y^3+py+q)=0 */
        coeffs[0]=q;
        coeffs[1]=p;
        coeffs[2]=0;
        coeffs[3]=1;
        num=SolveCubic(coeffs,s);
        s[num++]=0;
    } else {
        /* solve the resolvent cubic ... */
        coeffs[0]=1.0/2*r*p-1.0/8*q*q;
        coeffs[1]=-r;
        coeffs[2]=-1.0/2*p;
        coeffs[3]=1;
        (void)SolveCubic(coeffs,s);
        /* ... and take the one real solution ... */
        z=s[0];
        /* ... to build two quadric equations */
        v=z*z-r;
        v=2*z-p;
        if(IsZero(u))
            u=0;
        else if(u>0)
            u=sqrt(u);
        else
            return 0;
        if(IsZero(v))
            v=0;
        else if(v>0)
            v=sqrt(v);
        else
            return 0;
        coeffs[0]=z-u;
        coeffs[1]=v;
        coeffs[2]=1;
        num=SolveQuadric(ceoffs,s);

        coeffs[0]=z+u;
        coeffs[1]=-v;
        coeffs[2]=1;
        num+=SolveQuadric(coeffs,s+num);
    }
    /* resubstitute */
    sub=1.0/4*A;
    for(i=0;i<num,i++)
        s[i]-=sub;
    return num;
}
```

0.3.8 Gaussian Elimination

```
double A[300][301]; //for a 300*300 matrix
double x[300];  //solution
int N,M;  //A is an m*n matrix

int Gauss_solve(){  //solve Ax=b, where b=A[][N]
    double co,mul,result,tmp;
    int row,col,p,i,j;
    row=col=0;
    while(col<N && row<M){
        p=row;
        for(i=row+1;i<M;i++){
            if(fabs(A[i][col])>fabs(A[p][col]))  //*
                p=i;
        }
        for(i=col;i<=N;i++)
            tmp=A[row][i],A[row][i]=A[p][i],A[p][i]=tmp;
        if(A[row][col]==0){  //*
            col++;
            continue;
        }
        for(i=0;i<M;i++){
            if(i==row)continue;
            co=A[i][col]/A[row][col];
            for(j=col;j<=N;j++)
                A[i][j]=A[i][j]-co*A[row][j];
        }
        for(j=col+1;j<=N;j++)
            A[row][j]=A[row][j]/A[row][col];
        A[row][col]=1;  //*
        row++;
        col++;
    }
    for(i=row;i<M;i++){
        if(A[i][N]!=0)  //*
            break;
    }
    if(i!=M)
        return 2;
    for(i=0;i<N;i++)
        x[i]=0;
    for(i=0;i<M;i++){
        for(j=0;j<N;j++){
            if(A[i][j]!=0){  //*
                x[j]=A[i][N];
                break;
            }
        }
    }
    if(N>row)
        return 1;
    else
        return 0;
}
```

0.4 Structures

0.4.1 Big Integer

```
#define lld "lld" //change this into "I64d" if using windows.
const int Size=100;
const long long D=1000000000ll;
char __tmp[(Size+5)*9];
class bigint
{
    void fix(){
        s[n]=0;
        for(int i=0;i<n;i++)
            s[i+1]+=s[i]/D,s[i]%=D;
        if(s[n])n++;
        s[n]=0;
    }
public:
    int n;
    long long s[Size];
    bigint(const char a[]){
        int i;
        for(i=0;a[i];i++);
        for(n=0;(i-=9)>=0;n++)
```

```
            sscanf(a+i,"%9"lld,s+n);
        sprintf(__tmp,"%%%d"lld,9+i);
        if(i+9>0)sscanf(a,__tmp,s+n++);
        s[n]=0;
    }
    bigint(const long long &a){s[1]=a/D,s[0]=a%D,s[2]=0,n=1+!!s[1];}
    bigint(const int &a){*this=0ll+a;}
    bigint(){*this=0;}
    const char *tostr()const{
        int i,j;
        char *t=__tmp;
        sprintf(t,"%lld",s[n-1]);
        for(i=0;t[i];i++);
        for(j=n-2;j>=0;j--,i+=9)
            sprintf(t+i,"%09lld",s[j]);
        return t;
    }
    bigint& operator+=(const bigint &t)
    {
        for(int i=n;i<=t.n;i++)s[i]=0;
        if(n<t.n)n=t.n;
        for(int i=0;i<t.n;i++)
            s[i]+=t.s[i];
        fix();
        return *this;
    }
    bigint& operator*=(const bigint &t)
    {
        bigint tmp=*this;
        const long long *a=tmp.s,*b=t.s;
        if(b==s)b=a;
        n+=t.n;
        for(int i=0;i<=n;i++)s[i]=0;
        for(int i=0;i<tmp.n;i++)
            for(int j=0;j<t.n;j++)
            {
                s[i+j]+=a[i]*b[j];
                s[i+j+1]+=s[i+j]/D,s[i+j]%=D;
            }
        while(n>1 && !s[n-1])n--;
        return *this;
    }
/*  inline void operator+=(const int &x)
    {
        s[0]+=x;
        fix();
    }
    inline void operator*=(const int &x)
    {
        for(int i=0;i<n;i++)
            s[i]*=x;
        fix();
    }*/
};
```

0.4.2 Fibonacci Heap

```
struct node{
    int d,v,dg,mark;
    node *left,*right,*parent,*child;
    bool operator<(node a){return d<a.d;}
    void move(node *ptr){
        left->right=right,right->left=left;
        left=ptr,right=ptr->right;
        left->right=this,right->left=this;
    }
    void link(node *p){
        parent=p,p->dg++;
        if(!p->child)p->child=this;
        move(p->child);
    }
    void cut(node *&min){
        if(!parent)return;
        parent->child=right;
        if(parent->child==this)parent->child=0;
        move(min);
        if(parent->mark)parent->cut(min);
        else parent->mark=1,parent->dg--;
        parent=0;
        if(*this<*min)min=this;
    }
    void decrease(int nd,node *&min){
        d=nd;
        if(parent){if(*this<*parent)cut(min);}
        else if(*this<*min)min=this;
    }
};
class Fheap{
    node *list[30];
    void bond(){
        int i,d;
        node *now,*next;
        for(i=0;i<30;i++)list[i]=0;
        min->parent=0,next=min->right;
        do{
            now=next,now->parent=0;
            next=now->right;
            d=now->dg;
            while(list[d]){
                if(*now<*list[d])list[d]->link(now);
                else now->link(list[d]),now=list[d];
                list[d++]=0;
            }
            list[d]=now;
        }while(now!=min && !min->parent);
        for(i=0;i<30;i++)
            if(list[i] && *list[i]<*min)
                min=list[i];
        while(min->parent)min=min->parent;
    }
public:
    int n;node *min;
    Fheap(){min=0,n=0;}
    node *insert(int d,int v){
        node *ptr=new node;
        *ptr=(node){d,v,0,0,ptr,ptr,0,0};
        if(n++,min)ptr->move(min);
        else min=ptr;
        if(*ptr<*min)min=ptr;
        return ptr;
    }
    node extractmin(){
        node tmp=*min,*ptr=min;
        if(min->child){
            min->right->left=min->child;
            min->child->right->left=min;
            ptr=min->right,min->right=min->child->right,
            min->child->right=ptr;
            min->child=0;
        }
        ptr=min,min=min->right;
        ptr->move(ptr);
        if(--n)bond();
        else min=0;
        delete ptr;
        return tmp;
    }
};
```

0.5 Strings

0.5.1 Knuth-Morris-Pratt algorithm

```
#include<cstdio>
main(){
    int i,j,p[1001]={0,0};
    char s1[1001],s2[1001];
    while(scanf("%s %s",s1,s2)>0){
        for(i=0;s1[i];i++);
        s1[i++]='$',s1[i]=0;
        for(i=1,j=0;s1[i];i++){
            while(j && s1[i]!=s1[j])j=p[j];
            if(s1[i]==s1[j])j++;
            p[i+1]=j;
        }
        for(i=j=0;s2[i];i++){
            while(j && s1[j]!=s2[i])j=p[j];
            if(s1[j]==s2[i])j++;
        }
        printf("%d\n",j);
    }
}
```

0.5.2 Extended KMP

```
#include<cstdio>
#include<cstring>
struct node{
    node *sub,*c[26];
    int b;
}*root,*ptr1[100],*ptr2[100];
int to[256],now[1001],next[101],c[101][101];
char s[1000][1001],p[100][101];
void clear(node **ptr){
    if(*ptr){
        for(int i=0;i<26;i++)
            clear(&(*ptr)->c[i]);
        delete *ptr;
        *ptr=NULL;
    }
}
int build(node *&ptr,char *a,int b){
    if(!ptr){
        ptr=new node;
        for(int i=0;i<26;i++)
            ptr->c[i]=NULL;
        ptr->b=-1;
    }
    if(*a) return build(ptr->c[to[*a]],a+1,b);
    if(ptr->b>=0) return ptr->b;
    return(ptr->b=b);
}
main(){
    node *ptr;
    int i,j,n,m,t,sum;
    for(i='a';i<='z';i++)
        to[i]=i-'a';
    scanf("%d",&t);
    while(t--){
        clear(&root);
        scanf("%d %d",&n,&m);
        for(i=0;i<n;i++)
            scanf("%s",s[i]);
        scanf("%d %d",&m,&j);
        for(i=0;i<m;i++){
            next[i]=0;
            scanf("%s",p[i]);
            for(j=0;j<=i;j++)
                c[i][j]=c[j][i]=!strcmp(p[i],p[j]);
            j=build(root,p[i],i);
        }
        for(i=1,j=0;i<m;i++){
            while(j>0 && !c[i][j])
                j=next[j];
            j+=c[i][j];
            next[i+1]=j;
        }
        for(i=0;i<m;i++){
            ptr1[i]=root->c[to[p[i][0]]];
            ptr2[i]=root;
            ptr1[i]->sub=ptr2[i];
        }
        for(j=1;p[0][j];j++)
            for(i=0;i<m;i++){
                ptr1[i]=ptr1[i]->c[to[p[i][j]]];
                while(ptr2[i]!=root && !ptr2[i]->c[to[p[i][j]]])
                    ptr2[i]=ptr2[i]->sub;
                if(ptr2[i]->c[to[p[i][j]]])
                    ptr2[i]=ptr2[i]->c[to[p[i][j]]];
                ptr1[i]->sub=ptr2[i];
                if(ptr2[i]->b>0)
                    ptr1[i]->b=ptr2[i]->b;
            }
        for(j=0;s[0][j];j++)
            now[j]=0;
        for(i=sum=0;i<n;i++)
            for(j=0,ptr=root;s[i][j];j++){
                while(ptr!=root && !ptr->c[to[s[i][j]]])
                    ptr=ptr->sub;
                if(ptr->c[to[s[i][j]]])
                    ptr=ptr->c[to[s[i][j]]];
                if(ptr->b<0){
                    now[j]=0;
                    continue;
                }
                while(now[j]>0 && !c[ptr->b][now[j]])
                    now[j]=next[now[j]];
                if(c[ptr->b][now[j]])
                    now[j]++;
                if(now[j]==m)
                    sum++;
            }
        printf("%d\n",sum);
    }
}
```

0.5.3 Suffix Array

```
const int N=800000;
void rsort(int x[],int t0[],int t1[],int n,int m)
{
    static int num[N];
    int i;
    for(i=0;i<=m;i++)num[i]=0;
    for(i=0;i<n;i++)num[x[t0[i]]+1]++;
    for(i=1;i<=m;i++)num[i]+=num[i-1];
    for(i=0;i<n;i++)t1[num[x[t0[i]]]++]=t0[i];
}
void suffixarray(int text[],int t[],int m)
{
    int i,j,k,n;
    static int s[N*2],tmp[N+2];
    for(n=0;text[n];n++)s[n]=text[n],t[n]=n;
    for(i=t[n]=n;i<n*2;i++)s[i]=0;
    for(i=1;i<n;i*=2)
    {
        rsort(s+i,t,tmp,n,m);
        rsort(s,tmp,t,n,m);
        for(j=0;j<n;j++)tmp[j]=s[j];
        for(j=0,m=1;j<n;j++)
        {
            s[t[j]]=m;
            if(tmp[t[j]]!=tmp[t[j+1]] || tmp[t[j]+i]!=tmp[t[j+1]+i])m++;
        }
    }
}
int n,s[N],t[N],d[N],x[N];
void go(int s[],int t[],int d[],int n)
{
    int i,j,k;
    for(i=0;i<n;i++)x[t[i]]=i;
    for(i=j=0;i<n;i++)
    {
        if(x[i]==n-1)
        {j=d[x[i]]=0;continue;}
        k=t[x[i]+1];
        while(s[i+j]==s[k+j])
            j++;
        d[x[i]]=j;
        j-=(j>0);
    }
}
```

0.6 Geometry

0.6.1 Min Ball

```
struct min_ball {
    point center;
    number radius2;
    void add(const point& p) {
        ps.push_back(p);
    }
    void compile() {
        m = 0;
        center.resize(dim, 0);
        radius2 = -1;
        make_ball(ps.end());
    }
    template <class OUT>
        void support(OUT out) {
            copy(ps.begin(), supp_end, out);
        }
    min_ball() {
        for (int i = 0; i <= dim; ++i) {
            c[i].resize(dim, 0);
            v[i].resize(dim, 0);
        }
    }
    private:
    list<point> ps;
    list<point>::iterator supp_end;
    int m;
    point v[dim+1], c[dim+1];
    number z[dim+1], r[dim+1];
    void pop() { --m; }
    void push(const point& p) {
        if (m == 0) {
            c[0] = p; r[0] = 0;
        } else {
            number e = dist2(p, c[m-1]) - r[m-1];
            point delta = p - c[0];
            v[m] = p - c[0];
            for (int i = 1; i < m; ++i)
                v[m] -= v[i] * dot(v[i], delta) / z[i];
            z[m] = dot(v[m], v[m]);
            c[m] = c[m-1] + e*v[m]/z[m]/2;
            r[m] = r[m-1] + e*e/z[m]/4;
        }
        center  = c[m];
        radius2 = r[m]; ++m;
    }
    void make_ball(list<point>::iterator i) {
        supp_end = ps.begin();
        if (m == dim + 1) return;
        for (list<point>::iterator k = ps.begin(); k != i; ) {
            list<point>::iterator j = k++;
            if (dist2(*j, center) > radius2) {
                push(*j); make_ball(j); pop();
                move_to_front(j);
            }
        }
    }
    void move_to_front(list<point>::iterator j) {
        if (supp_end == j) ++supp_end;
        ps.splice (ps.begin(), ps, j);
    }
};
```

0.6.2 Segment Intersection

```
inline int one(int x){return x<0?-1:x>0?1:0;}
bool intersect(int e,int f,int g,int h,int o,int p,int q,int r){
    if((e>?g)<(o<?q))return 0;
    if((e<?g)>(o>?q))return 0;
    if((f>?h)<(p<?r))return 0;
    if((f<?h)>(p>?r))return 0;
    int s=one((e-o)*(h-p)-(g-o)*(f-p)),
    t=one((e-q)*(h-r)-(g-q)*(f-r)),
    u=one((o-e)*(r-f)-(q-e)*(p-f)),
    v=one((o-g)*(r-h)-(q-g)*(p-h));
    if(s*t<=0&&v*u<=0)return 1;
    return 0;
}
```

0.6.3 Convex Polygon Intersection

```
#include<stdio.h>
double abs(double n){
    return n<0?-n:n;
}
struct pt{
    double x,y;
    void get(){
        scanf("%lf %lf",&x,&y);
    }
    int operator==(pt s){
        return(abs(x-s.x)<1e-6 && abs(y-s.y)<1e-6);
    }
}p1[100],p2[100],tmp[1000],cov[1000],_;
int cn;
double dis(pt a,pt b){
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}
double rate(pt o,pt a,pt b){
    return (a.x-o.x)*(b.y-o.y)-(b.x-o.x)*(a.y-o.y);
}
int atline(pt a,pt b,pt p){
    if(abs(rate(a,b,p))<1e-6&&dis(a,b)>=dis(a,p)&&dis(a,b)>=dis(b,p))
        return 1;
    return 0;
}
double area(pt *v,int n){
int i;
double s=0.0;
    for(i=0;i<n;i++)
        s+=v[i].x*v[i+1].y-v[i].y*v[i+1].x;
    return abs(s);
}
double area(pt a,pt b,pt c){
    if(a==b || a==c || b==c)return 0;
    return abs(a.x*b.y+b.x*c.y+c.x*a.y-a.x*c.y-b.x*a.y-c.x*b.y);
}
void insert(pt p1,pt p2,pt q1,pt q2){
pt p;
double a1=p2.y-p1.y,b1=p1.x-p2.x,c1=p1.x*a1+p1.y*b1,
      a2=q2.y-q1.y,b2=q1.x-q2.x,c2=q1.x*a2+q1.y*b2;
    if(a1*b2==a2*b1)return;
    p.x=(c1*b2-c2*b1)/(a1*b2-a2*b1);
    p.y=(a1*c2-a2*c1)/(a1*b2-a2*b1);
    if(atline(p1,p2,p) && atline(q1,q2,p))
        tmp[cn++]=p;
}
void inpoly(pt *v,int n,double parea,pt p){
int i;
    for(i=0;i<n;i++)
        parea-=area(p,v[i],v[i+1]);
    if(parea>=0.0)tmp[cn++]=p;
}
main(){
int i,j,n1,n2,min;
double a1,a2,ans;
while(scanf("%d",&n1) && n1){
    for(cn=i=0;i<n1;i++)p1[i].get();
    p1[n1]=p1[0];
    scanf("%d",&n2);
    for(i=0;i<n2;i++)p2[i].get();
    p2[n2]=p2[0];
    for(i=1;i<=n1;i++)
        for(j=1;j<=n2;j++)
            insert(p1[i-1],p1[i],p2[j-1],p2[j]);
    a1=area(p1,n1);
    a2=area(p2,n2);
    ans=a1+a2;
    for(i=0;i<n1;i++)
        inpoly(p2,n2,a2,p1[i]);
    for(i=0;i<n2;i++)
        inpoly(p1,n1,a1,p2[i]);
    for(min=i=0;i<cn;i++)
        if(tmp[i].x<tmp[min].x ||
        (tmp[i].x==tmp[min].x && tmp[i].y<tmp[min].y))min=i;
    _=tmp[0],tmp[0]=tmp[min],tmp[min]=_;
    for(j=0;j<cn;j++)
        for(i=j+1;i<cn;i++)
            if(tmp[i]==tmp[j]){
                _=tmp[cn-1],tmp[cn-1]=tmp[i],tmp[i]=_;
                cn--;
            }
    for(i=1;i<cn;i++)
        for(j=2;j<cn;j++)
            if(rate(tmp[0],tmp[j],tmp[j-1])<0 ||
            (rate==0 && dis(tmp[0],tmp[j-1])>dis(tmp[0],tmp[j]))){
                _=tmp[j],tmp[j]=tmp[j-1],tmp[j-1]=_;
            }
    cov[0]=tmp[--cn];
    for(i=1;cn--;i++){
        while(cn>=0 && cov[i-1]==tmp[cn])cn--;
        if(cn<0)break;
        cov[i]=tmp[cn];
    }
    cov[i]=cov[0];
    ans-=2*area(cov,i);
    printf("%8.2lf",ans/2);
}
puts("");
}
```

0.6.4 Union area

```
/* $Id: rect.c,v 1.1 2002/02/25 09:46:50 kcwu Exp $ */
/*from USACO 1.2.1 solution*/
#include <stdio.h>
#inculde <stdlib.h>
#inculde <string.h>
FILE *fp,*fo;
struct rect
{
    int c;
    int x1,y1,x2,y2;
};
int c[2501];
rect r[10001];
int intersect(rect a, const rect &b, rect out[4])
{
    /* do they at all intersect? */
    if(b.x2<a.x1||b.x1>=a.x2)
        return 0;
    if(b.y2<a.y1||b.y1>=a.y2)
        return 0;
    /* they do*/
    rect t;
    if(b.x1<=a.x1&&b.x2>=a.x2&&b.y1<=a.y1&&b.y2>=a.y2)
        return -1;
    /* cutting ‘a’ down to match b */
    int nout=0;
    if(b.x1>=a.x1) {
        t=a,t.x2=b.x1;
        if(t.x1!=t.x2)
            out[nout++]=t;
        a.x1=b.x1;
    }
    if(b.x2<a.x2) {
        t=a,t.x1=b.x2;
        if(t.x1!=t.x2)
            out[nout++]=t;
        a.x2=b.x2;
    }
    if(b.y1>=a.y2) {
        t=a,t.y2=b.y1;
        if(t.y1!=t.y2)
            out[nout++]=t;
        a.y1=b.y1;
    }
    if(b.y2<a.y2) {
        t=a,t.y1=b.y2;
        if(t.y1!=t.y2)
            out[nout++]=;
              a.y2=b.y2;
    }
    return nout;
}
int main(void) {
    int a,b,n;
    fscanf(fp,"%d %d %d", &a, &b, &n);
    r[0].c=1;
    r[0].x1=r[0].y1=0;
    r[0].x2=a;
    r[0].y2=b;
    rect t[4];
    int i,j,rr=1;
    for(i=0;i<n;i++) {
        int tmp;
        fscanf(fp,
        "%d%d%d%d%d",&r[rr].x1,&r[rr].y1,&r[rr].x2,&r[rr].y2,&r[rr].c);

        if(r[rr].x1>r[rr].x2) {
            tmp=r[rr].x1;
            r[rr].x1=r[rr].x2;
            r[rr.x2=tmp;
        }
        if(r[rr.y1>r[rr].y2) {
            tmp=r[rr].y1;
            r[rr].y1=r[rr].y2;
            r[rr].y2=tmp;
        }
        int nr=rr;
        rect curr=r[rr++];
        for(j=0;j<nr;j++) {
            int n=intersect(r[j],curr,t);
            if(!n)
                continue;
            if(n==-1) {
                memmove(r+j,r+j+1,sizeof(rect)*(rr-j-1));
                j--;
                rr--;
                nr--;
                continue;
            }
            r[j]=t[--n];
            for(;n-->0;)
                r[rr++]=t[n];
        }
    }
    for(i=0;i<rr;i++)
        c[r[i].c]+=(r[i].x2-r[i].x1)*(r[i].y2-r[i].y1);
    for(i=1,i<=250;i++)
        if(c[i])
            fprintf(fo,"%d %d\n",i,c[i]);
    return 0;
}
```

0.6.5 Intersection

```
/* $Id: inclustion_exclusion.c,v 1.1 2002/02/25 09:46:50 kcwu Exp $ */
#include <stdio.h>
#define MAX(a,b) ((a)>(x)?(a):(b))
#define MIN(a,b) ((a)<(b)?(a):(b))
#define MAX_T 10000
struct square {
    int sign;
    double x,X,y,Y;
} antenna[MAX_T];

int main(void)
{
    int i,j;
    int ncase=0;
    double x,y,r;
    int n,t,append;
    struct square tmp,over;
    double area;
    while(scanf("%d",&n)==1 && n) {
        t=0;
        for(i=0;i<n;i++) {
            scanf("%lf%lf%lf", &x, &y, &r);
            tmp.x=x-r; tmp.y=y-r; tmp.X=x+r; tmp.Y=y+r;
            tmp.sign=1;
            append=t;
            antenna[append++]=tmp;
            for(j=0;j<t;j++) {
                over.x=MAX(antenna[j].x,tmp.x);
                over.y=MAX(antenna[j].y,tmp.y);
                over.X=MIN(antenna[j].X,tmp.X);
                over.Y=MIN(antenna[j].Y,tmp.Y);
                over.sign=-antenna[j].sign*tmp.sign;
                if((over.X>over.x && over.Y>=over.y) ||
                   (over.X==over.x && over.Y>over.y)) {
                    if(over.X==tmp.X && over.x==tmp.x &&
                       over.Y==tmp.Y && over.y==tmp.y) {
                        append=t;
                        break;
                    } else
                        antenna[append++]=over;
                }
            }
            t=append;
        }
        area=0;
        for(i=0;i<t;i++)
            area+=antenna[i].sign*(antenna[i].X-antenna[i].x)
                                 *(antenna[i].Y-antenna[i].y);
        printf("%d %.2f\n",++ncase,area);
    }
    return 0;
}
```

0.6.6 Convex Hull

```
#include<cstdio>
#include<algorithm>
struct pt{
   int x,y;
   pt operator-(pt a)const{return(pt){x-a.x,y-a.y};}
   int operator*(pt a){return x*a.y-y*a.x;}
}s[2009];
int dis(pt a,pt b)
{return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);}
bool ptcmp(const pt& a,const pt& b)
{return(a-s[0])*(b-s[0])<0;}
int t[2009];
main(){
   int i,j,k,n,tmp,T;
   scanf("%d",&T),printf("%d\n",T);
   while(T--){
       scanf("%d",&n);
       for(i=j=0;i<n;i++){
           scanf("%d %d",&s[i].y,&s[i].x);
           if(s[i].x<s[j].x ||(s[i].x==s[j].x && s[i].y<s[j].y))j=i;
       }
       s[n-1]=s[0];
       s[2008]=s[0],s[0]=s[j],s[j]=s[2008];
       std::sort(s+1,s+n,ptcmp);
       s[n++]=s[0];
       t[0]=0,t[1]=1;
       for(i=2,k=2;i<n;i++){
           tmp=(s[i%n]-s[t[k-2]])*(s[t[k-1]]-s[t[k-2]]);
           if(tmp==0){
               tmp=dis(s[t[k-2]],s[t[k-1]])-dis(s[t[k-2]],s[i]);
               if(tmp>0)continue;
           }
           while(k>1 && tmp<=0)
               k--,tmp=(s[i%n]-s[t[k-2]])*(s[t[k-1]]-s[t[k-2]]);
           t[k++]=i%n;
       }
       printf("%d\n",k);
       for(i=0;i<k;i++)
           printf("%d %d\n",s[t[i]].y,s[t[i]].x);
       if(T)scanf("%d",&i),puts("-1");
   }
}
```

0.7 Misc.

0.7.1 Poker

```
inline int rank(unsigned x)
{
    int i,t=0;
    for(i=7311616;i>0;i/=52)
        t=t*13+(((x/i)%52)/4);
    return t;
}
struct hand
{
    unsigned h;
    hand(){}
    bool operator<(hand a)const{
        if(h/380204032!=a.h/380204032)
            return h/380204032<a.h/380204032;
        return rank(h%380204032)<rank(a.h%380204032);
    }
    bool operator==(hand a)const{
        if(h/380204032!=a.h/380204032)
            return 0;
        return rank(h%380204032)==rank(a.h%380204032);
    }
    hand(int a,int b){h=a+b*380204032ll;}
    void print()
    {
        unsigned i,j;
        for(i=0,j=h;i<5;i++,j/=52)
            printf("%c%c ",rk[j%52/4],su[j%52%4]);
        puts("");
    }
}s0[2598920],*s1=s0+1296420;
inline int test(int card[])
{
    char c[52],s[4],x[26];
    fill(s,s+4,0);
    fill(x,x+26,0);
    fill(c,c+52,0);
    int i,j,y,z;
    for(i=0;i<m;i++)
        s[card[i]%4]++,x[card[i]/4]++,c[card[i]]++;
    char k=0,f=0,S=0,t=0,tp=0,p=0;
    for(i=0;i<13;i++)
    {
        if(x[i]==4)k=1;
        if(x[i]==3)t=1;
        if(x[i]==2)
        {
            if(!p)p=1;
            else tp=1;
        }
        x[i+13]=x[i];
    }
    for(i=0;i<4;i++)
        if(s[i]>=5)f=1;
    for(i=0;i<9;i++)
        if(x[i] && x[i+1] && x[i+2] && x[i+3] && x[i+4])
            S=1;
    if(x[12] && x[0] && x[1] && x[2] && x[3])
        S=1;
    j=0;
    if(p)j=1;
    if(tp)j=2;
    if(t)j=3;
    if(S)j=4;
    if(f)j=5;
    if(t && p)j=6;
    if(k)j=7;
    if(f && S)
    {
        for(i=z=0;z<5 && i<52;i++)
        {
            y=i/13+(i%13)*4;
            if(y/4==0)z=0;
            if(c[y])z++;
            else z=0;
            if(y/4==3 && c[y%4+48])z++;
        }
        if(z>=5)j=8,print(card);

    }
    printf("%d ",j);
    return j;
}
```

0.7.2 Stable Matching

```
vector<int> stable_matching(const vector<vector<int> >& orderM,
        const vector<vector<int> >& orderW) {
    const int N = orderM.size();
    vector< vector<int> > preferW(N, vector<int>(N+1, N));
    vector<int> matchW(N, N), proposedM(N);
    for (int w = 0; w < N; ++w)
        for (int i = 0; i < N; ++i)
            preferW[w][ orderW[w][i] ] = i;
    for (int m_ = 0; m_ < N; ++m_) {
        for (int m = m_; m < N; ) {
            int w = orderM[m][ proposedM[m]++ ];
            if (preferW[w][ m ] < preferW[w][ matchW[w] ])
                swap(m, matchW[w]);
        }
    }
    return matchW;
}
```

0.7.3 Decompression

```
#include<cstdio>
#include<algorithm>
using namespace std;
char s0[1000005],s1[1000005];
int c[1000005];
int dec(char a[],char b[])
{
    int i,k,n=0;
    for(i=0;a[i];i++)
    {
        if('0'<=a[i] && a[i]<='9')
        {
            for(k=0;'0'<=a[i] && a[i]<='9';i++)
                k=k*10+a[i]-'0';
            for(i--,k--;k--;)
                b[n]=b[n++-1];
        }
        else b[n++]=a[i];
    }
    b[n]=0;
    return n;
}
main()
{
    int i,j,n,T;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%s",s0);
        for(n=0;s0[n];n++);
        copy(s0,s0+n+1,s1);
        n=dec(s1,s0);
        copy(s0,s0+n+1,s1);
        sort(s1,s1+n);
        for(i=j=0;i<n;i++)
        {
            if(i && s1[i]!=s1[i-1])j=0;
            for(;s0[j]!=s1[i];j++)
                if(j>=n)puts("!!");
            c[i]=j++;
        }
        for(i=c[c[0]];i!=c[0];i=c[i])
            putchar(s0[i]);
        puts(".");
    }
}
```

0.7.4 Longest Palindrome

```
int longest_palindrome(char *text, int n) {
    int rad[2*n], i, j, k;
    for (i = 0, j = 0; i < 2*n; i += k, j = max(j-k, 0)) {
        while (i-j >= 0 && i+j+1 < 2*n && text[(i-j)/2]==text[(i+j+1)/2])++j;
        rad[i] = j;
        for (k = 1; i-k >= 0 && rad[i]-k >= 0 && rad[i-k] != rad[i]-k; ++k)
            rad[i+k] = min(rad[i-k], rad[i]-k);
    }
    return *max_element(rad, rad+2*n); // ret. centre of the longest palindrom
}
```

0.7.5 Date

```
int tonum(int y,int m,int d){
    static int day[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
    y--;
    int num=y*365+y/4-y/100+y/400;
    y++;
    if((y%4==0 && y%100)|| y%400==0)day[2]=29;
    else day[2]=28;
    for(int i=1;i<m;i++)
        num+=day[i];
    return num+=d;
}
void todate(int num,int &y,int &m,int &d){
    static int y400=146097,y100=36524,y4=1461,
        day[]={0,31,28,31,30,31,30,31,31,30,31,30,31};;
    num--;
    y=num/y400*400;
    num%=y400;
    y+=num/y100*100;
    num%=y100;
    y+=num/y4*4;
    num%=y4;
    y+=num/365;
    num%=365;
    for(m=1;num>=day[m];num-=day[m++]);
    d=++num;
}
```