#include <iostream>

using namespace std;

typedef long long ll;

struct T{};

int in()

{

    int c=getchar(),x=0,f=1;

    while (!isdigit(c) && c!='-') c=getchar();

    if (c=='-') c=getchar(),f=-1;

    while (isdigit(c)) x=x\*10+c-'0',c=getchar();

    return x\*f;

}

int qrand(){

    static int seed=2333;

    return seed = (int)((((seed ^ 998244353)+19260817ll)\*12345678ll)%1000000007);

}

T gcd(T a, T b){ return b==0?a:gcd(b, a%b);}

int gcd(int a, int b) {return b?gcd(b,a%b):a;}

ll qmul(ll x,ll y,ll p){

    ll t=(x\*y-(ll)((long double)x/p\*y+1.0e-8)\*p);

    return t<0 ? t+p : t;

}

//return a^x

T qpow(T a, int x){

    T ans=1;

    for (;x;a\*=a,x>>=1)

        if (x&1)

            ans\*=a;

    return ans;

}

ll qpow(ll a, int x){

    T ans=1;

    for (;x;a\*=a,x>>=1)

        if (x&1)

            ans\*=a;

    return ans;

}

ll qpow(ll a, int x, ll p){

    ll ans=1;

    for (;x;a=qmul(a,a,p),x>>=1)

        if (x&1)

            ans=qmul(ans,a,p);

    return ans;

}

const int N;

struct Mat{

    ll m[N][N];

    Mat(){memset(m,0,sizoef(m));}

    I(){for (int i=0;i<n;i++) m[i][i]=0;}

}

Mat mul(const &Mat a, const &Mat b){

    Mat c;

    for (int i=0;i<N;i++)

        for (int j=0;j<N;j++){

            for (int k=0;k<=N;k++)

                c.m[i][j]+=a.m[i][j]\*b.m[i][j];

            //c.m[i][j]%=p;

        }

}

Mat qpow(Mat a, int x){

    Mat ans; ans.I();

    for (;x;a=mul(a,a),x>>=1)

        if (x&1)

            ans=mul(ans,a);

    return ans;

}

const int maxn;

int p[maxn],phi[maxn],pc;

bool isp[maxn];

void gen(){

    memset(isp,1,sizeof(isp));

    isp[1]=0;

    for (int i=2;i<maxn;i++){

        if (isp[i]) p[pc++]=i,phi[i]=i-1;

        for (int j=0;j<pc && i\*p[j]<maxn;j++){

            isp[i\*p[j]]=0;

            if (i%p[j]==0){

                phi[i\*p[j]]=phi[i]\*p[j];

                break;

            }

            phi[i\*p[j]]=phi[i]\*(p[j]-1);

        }

    }

}

int inv(int x, int p){

    return qpow(x,p-2,p);

}

//ax+by=gcd(a,b)

int exgcd(int a, int b, int &x, int &y){

    if (b==0){

        x=1;y=0;

        return a;

    }

    int t=exgcd(b,a%b,y,x);

    y-=a/b\*x;

    return t;

}

int inv(int v, int p){

    int x,y;

    if (exgcd(v,p,x,y)==1)

        return (x+p)%p;

    else

        return -1;

}

//中国剩余定理

//x=a1(mod p1)

//x=a2(mod p2)

//...

ll china(int n, ll a[], ll p[]){

    ll M=1,x=0;

    for (int i=0;i<n;i++) M\*=p[i];

    for (int i=0;i<n;i++)   {

        ll w=M/m[i]; //x=pi\*k1+a + w\*k2

        x=(x+w\*inv(m[i],w)\*a[i])%M; //k1 pi\*k1=a (Mod w)

    }

    return (x+M)%M;

}

//中国剩余定理\_非互质

//x=a1(mod m1)

//x=a2(mod m2)

//...

ll china1(int n, ll a[], ll p[]){

    ll n1=a[0],a1=a[0],n2,a2,k1,k2,K,gcd,c,t;

    for (int i=1;i<n;i++){//依次合并方程

        n2=n[i],a2=a[i];

        c=a2-a1;

        gcd=exgcd(n1,n2,k1,k2); //n1\*k1+n2\*k2=gcd(n1,n2)

        if (c%gcd) return -1;

        K=c/gcd\*k1;//n1\*K+n2\*(c/gcd\*k2)=c

        t=n2/gcd; K=(K+t)%t; //K>=0

        a1+=n1\*K;

        n1\*=n2/gcd;

    }

    return a1;

}

//discrete logarithm

//a^x=b(mod p)

ll BSGS(ll a, ll b, ll p){

    int m,v,e=1,i;

    m=(int)sqrt(p+0.5);

    v=inv(pow(a,m,p),p);

    map<int,int> x; //hash\_map -> O(sqrt(N))

    x[1]=0;

    for (int i=1;i<m;i++){

        e=e\*a%p;

        if (!x.count(e))

            x[e]=i;

    }

    for (int i=0;i<m;i++){

        if (x.count(b))

            return i\*m+x[b];

        b=b\*v%p;

    }

    return -1;

}

const ll p0[]={2,3,5,7,11,13,17,19,23,29,31};

//a^(p-1)=1 (mod p) , x^2=1 (mod p) while x=1 or p-1

bool witness(ll a,ll b,ll r,ll s){

    ll x=qpow(a,r,n),pre=x;

    for (int i=0;i<s;i++){

        x=qmul(x,x,n);

        if (x==1 && pre!=1 && pre!=n-1) return 0;

        pre=x;

    }

    if (x!=1) return 0;

    return 1;

}

bool MillerRabin(ll n){

    if (n<=1) return 0;

    ll r=n-1,s=0;

    while (!(r&1)) r>>=1,s++;

    for (int i=0;i<10;i++){

        if (p0[i]==n) return 1;

        if (!check(p0[i],n,r,s)) return 0;

    }

    return 1;

}

ll pol\_rho(ll n,ll c){

    ll k=2,x=rand()%n,y=x,p=1;

    for (ll i=1;p==1;i++){

        x=(qmul(x,x,n)+c)%n;

        p=y>x?y-x:x-y;

        p=gcd(n,p);

        if (i==k)

            y=x,k+=k;

    }

    return p;

}

void spiltprime(ll n)

{

    if (n==1) return;

    if (MillerRabin(n)) {maxs=max(maxs,n); return;} //n is prime factor

    ll t=n;

    while (t==n) t=pol\_rho(n,rand()%(n-1));

    spiltprime(t); spiltprime(n/t);

}

namespace LUCAS{

const ll luo=10007;

ll fact[luo],vfact[luo];

ll comb(ll n,ll m){

    if (n<m) return 0;

    return fact[n]\*vfact[n-m]%luo\*vfact[m]%luo;

}

ll lucas(ll n,ll m){

    if (m==0) return 1;

    return lucas(n/luo,m/luo)\*comb(n%luo,m%luo)%luo;

}

void pre(){

    fact[0]=1;

    for (int i=1;i<luo;i++) fact[i]=fact[i-1]\*i%luo;

    for (int i=0;i<luo;i++) vfact[i]=qpow(fact[i], luo-2, luo);

}

}

namespace Graph{

const int maxn=10010,maxm=100010,inf=0x3f3f3f3f;

int head[maxn],nxt[maxm],to[maxm],co[maxm],ec;

int n;

bool vis[maxn];

int ans[maxn][maxn],dis[maxn],c[maxn];

void added(int x, int y, int c){

    ec++;

    nxt[ec]=head[x];

    head[x]=ec;

    to[ec]=y;

    co[ec]=c;

}

int spfa(int s){

    queue<int> q;

    memset(dis,0x3f,sizeof(dis));

    dis[s]=0;

    memset(c,0,sizeof(c)); //判负环

    memset(vis,0,sizeof(vis));

    q.push(s); vis[s]=c[s]=1;

    while (!q.empty())  {

        int u=q.front(); q.pop();

        vis[x]=0;

        for (int e=head[x];e;e=nxt[e]){

            int v=to[k];

            if (dis[u]+co[k]<dis[v]){

                dis[v]=dis[u]+co[i];

                if (!vis[v]){

                    vis[v]=1;

                    c[v]++;

                    q.push(v);

                    if (c[v]]>n) return -0x3f3f3f3f;

                }

            }

        }

    }

}

//judge negative circle

bool spfa\_dfsjudge(int u){

    vis[u]=1;

    for (int e=head[u];e;e=nxt[e]){

        int v=to[e];

        if (dis[u]+co[e]<dis[v]){

            dis[v]=dis[u]+co[e];

            if (vis[v] || spfa\_dfsjudge(v)) return true;

        }

    }

    vis[u]=0;

    return false;

}

void dijk(int s){

    memset(dis,0x3f,sizeof(dis));

    memset(vis,0,sizeof(vis));

    dis[s]=0;

    priority\_queue<pair<int,int>> qu;

    qu.push(make\_pair(0,s))

    while (qu.size()){

        int u=qu.top().second, mc=-qu.top().first;

        qu.pop();

        if (vis[u]) continue;

        vis[u]=1;

        for (int e=head[u];e;e=nxt[e])

            if (!vis[to[e]] && mc+co[e]<dis[to[e]]){

                dis[to[e]]=mc+co[e];

                qu.push(make\_pair(-dis[to[e]],to[e]))

            }

    }

}

void floyd(){

    for (int k=0;k<n;k++)

        for (int i=0;i<n;i++)

            for (int j=0;j<n;j++)

                if (d[i][j]>d[i][k]+d[k][j])

                    d[i][j]=d[i][k]+d[k][j];

}

//mp为原图，求无向图最小环。有向图为d[i][i]。

int floyd\_minc(){

    int minc=inf;

    for (int k=0;k<n;k++){

        for (int i=0;i<k;i++)

            for (int j=i+1;j<k;j++)

                minc=min(minc,d[i][j]+mp[i][k]+mp[k][j]);

        for (int i=0;i<n;i++)

            for (int j=0;j<n;j++)

                if (d[i][j]>d[i][k]+d[k][j])

                    d[i][j]=d[i][k]+d[k][j];

    }

    return minc;

}

vector<int> ed[maxn];

bool ins[maxn];

int st[maxn],stn[maxn],low[maxn],idx,scn;

vector<int> scc[maxn];

void tarjan(int u){

    st[stn++]=u;

    ins[u]=1;

    dfn[u]=low[u]=++idx;

    for (int i=0;i<ed[u].size();i++)

        if (!dfn[ed[u][i]]){

            tarjan(ed[u][i]);

            low[u]=min(low[u],low[ed[u][i]]);

        }

        else if (ins[ed[u][i]])

            low[u]=min(low[u],low[ed[u][i]]);

    if (low[u]==dfn[u]){

        int v;

        do{

            v=st[--stn];

            scc[scn].push\_back(v);

            ins[v]=0;

        }while (u!=v);

    }

}

void tarjan\_Caller(){

    for (int i=1;i<=n;i++)

        if (!dfn[i])

            tarjan(i);

}

bool iscut[maxn];

void tarjan\_point(int u, int fa){

    int ch=0;

    low[u]=dfn[u]=++idx;

    for (int i=0;i<ed[u].size();i++){

        int v=ed[u][i];

        if (!dfn[v]){

            ch++;

            tarjan\_point(v,u);

            low[u]=min(low[u],low[v]);

            if (low[v]>=dfn[u]) iscut[u]=1;

        }

        else if (dfn[v]<dfn[u] && v!=fa)

            low[u]=min(low[u],dfn[v]);

    }

    if (fa<0 && ch==1) iscut[u]=0;

}

void tarjan\_ed(int u, int fa){

    low[u]=dfn[u]=++idx;

    for (int i=0;i<ed[u].size();i++){

        int v=vec[u][i];

        if (!dfn[v]){

            tarjan(v,u);

            low[u]=min(low[u],low[v]);

            if (low[v]>dfn[u])

                ansx[ansc]=x,ansy[ansc++]=y;

        }

        else if (dfn[v]<dfn[u] && v!=fa)

            low[u]=min(low[u],dfn[v]);

    }

}

}

namespace FFT

{

typedef complex<double> cd;

const int maxl=(1<<20)+1,pi=3.14159265358979;

cd a[maxl],b[maxl];

int rev[maxl];

void get\_rev(int bit){

    for (int i=0;i<(1<<bit);i++)

        rev[i]=(rev[i>>1]>>1)|((i&1)<<(bit-1));

}

void fft(cd a[], int n, int dft){

    for(int i=0;i<n;i++) if(i<rev[i]) swap(a[i],a[rev[i]]);

    for (int s=1;s<n;s<<=1){

        cd wn=exp(cd(0,pi\*dft/s));

        for (int j=0;j<n;j+=s<<1){

            cd wnk(1,0);

            for (int k=j;k<j+s;k++){

                cd x=a[k],y=wnk\*a[k+s];

                a[k]=x+y;

                a[k+s]=x-y;

                wnk\*=wn;

            }

        }

    }

    if (dft==-1) for (int i=0;i<n;i++) a[i]/=n;

}

char s1[maxl],s2[maxl];

int ans[maxl];

void mul(){

    scanf("%s%s",s1,s2);

    int l1=strlen(s1),l2=strlen(s2);

    int s=2,bit=1;

    for (bit=1;(1<<bit)<l1+l2-1;bit++)s<<=1;

    for (int i=0;i<l1;i++) a[i]=s1[l1-i-1]-'0';

    for (int i=0;i<l2;i++) b[i]=s2[l2-i-1]-'0';

    get\_rev(bit);

    fft(a,s,1); fft(b,s,1);

    for (int i=0;i<s;i++) a[i]\*=b[i];

    fft(a,s,-1);

    for (int i=0;i<s;i++){

        ans[i]+=(int)(a[i].real()+0.5);

        ans[i+1]+=output[i]/10;

        ans[i]%=10;

    }

    int i;

    for (i=l1+l2;!ans[i]&&i>=0;i--);

    if (i==-1) printf("0");

    for (;i>=0;i--) printf("%d",ans[i]);

    putchar('\n');

}

}

namespace Expr{

//Easy experission, calc +-\*/^()

#define CP cin.peek()

#define CG cin.get()

#define CS while (CP==' ') CG;

int S();

int V(){CS

    int ans=0;

    if (CP=='('){

        CG;

        ans=S();

        CS;CG;

    }

    else cin>>ans;

    return ans;

}

int U(){

    int ans=V(); CS;

    while (CP=='^'){

        CG;

        int v=V(),d=ans;

        if (v==0) ans=1;

        for (int i=1;i<v;i++)

            ans\*=d;

    }

    return ans;

}

int T(){

    int ans=U(); CS;

    while (CP=='\*' || CP=='/'){

        if (CG=='\*') ans\*=U();

        else ans/=U();

    }

    return ans;

}

int S(){

    int ans=0; CS;

    if (CP=='-'){

        CG; ans=-T();

    }

    else ans=T();

    CS;

    while (CP=='+' || CP=='-'){

        if (CG=='+') ans+=T();

        else ans-=T();

    }

    return ans;

}

#undef CG

#undef CP

#undef CS

}

namespace UFSet{

    const int maxn=100010;

    int fa[maxn];

    void clear(){

        for (int i=0;i<maxn;i++) fa[i]=i;

    }

    void fi(int x){

        if (fa[x]!=x)

            fa[x]=fi(fa[x]);

        return fa[x];

    }

    void un(int a, int b){

        int ta=fi(a),tb=fi(b);

        if (ta!=tb) fa[ta]=tb;

    }

}

namespace TreeArr{

#define lowbit(x) (x&(-x))

const int maxn=100010;

ll tr[maxn]; int n;

ll sum(int x){

    ll ret=0;

    while (x){

        ret+=tr[x];

        x-=lowbit(x);

    }

    return ret;

}

void add(ll a, int x){

    while (x<=n){

        tr[x]+=a;

        x+=lowbit(x);

    }

}

}

}

namespace BipartiteGraph{

int vis[maxn]; //memset to 0 when start

//judge if a map is BipartiteGraph

bool judge(int u, int col){

    vis[u]=col;

    for (int i=0;i<n;i++)

        if (d[u][i] && (vis[i] && vis[i]!=-col || !vis[i] && !judge(i,-col))

            return 0;

    return 1;

}

const int maxn=500;

//to: m->n

int d[maxn][maxn],to[maxn],n,m;

bool vis[maxn];

bool xiong(int u){

    vis[u]=1;

    for (int i=0;i<m;i++)

        if (d[u][i] && !vis[i]){

            vis[i]=1;

            if (!to[i] || xiong(to[i])){

                to[i]=u;

                return 1;

            }

        }

    return 0;

}

int match(){

    int ans=0;

    for (int i=0;i<n;i++){

        memset(vis,0,sizeof(vis));

        if (Xiong(i)) ans++;

    }

    return ans;

}

}

namespace SEGT{

const int MAXN=100010;

ll sum[MAXN<<2], tadd[MAXN<<2], tmul[MAXN<<2], a[MAXN];

ll n,m,p;

#define lc u+u+1

#define rc u+u+2

void build(int u, int l, int r){

    tmul[u]=1;

    if (l==r-1){

        sum[u]=a[l];

        return;

    }

    int mid=l+r>>1;

    bulid(lc,l,mid); build(rc,mid,r);

    sum[u]=(sum[u+u+1]+sum[u+u+2])%p;

}

void upd(int u, int l, int r){

    int mid=l+r>>1;

    sum[lc]\*=tmul[u]; sum[lc]+=(mid-l)\*tadd[u]; sum[lc]%=p;

    sum[rc]\*=tmul[u]; sym[rc]+=(r-mid)\*tadd[u]; sum[rc]%=p;

    tadd[lc]\*=tmul[u]; tadd[lc]+=tadd[u]; tadd[lc]%=p;

    tmul[lc]\*=tmul[u]; tmul[lc]%=p;

    tadd[rc]\*=tmul[u]; tadd[rc]+=tadd[u]; tadd[rc]%=p;

    tmul[rc]\*=tmul[u]; tmul[rc]%=p;

    tadd[u]=0; tmul[u]=1;

}

void mul(int u, int l, int r, int cl, int cr, ll c){

    if (cl<=l && cr>=r){

        tadd[u]\*=c; tadd[u]%=p;

        tmul[u]\*=c; tmul[u]%=p;

        sum[u]\*=c; sum[u]%=p;

        return;

    }

    if (tadd[u] || tmul[u]!=1) upd(u,l,r);

    int mid=l+r>>1;

    if (cl<mid) mul(lc,l,mid,cl,cr,c);

    if (cr>mid) mul(rc,mid,r,ck,cr,c);

    sum[u]=(sum[lc]+sum[rc])%p;

}

void add(int u, int l, int r, int cl, int cr, ll c){

    if (cl<=l && cr>=r){

        tadd[u]+=c; tadd[u]%=p;

        sum[u]=c\*(r-l)%p; sum[u]%=p;

        return;

    }

    if (tadd[u] || tmul[u]!=1) upd(u,l,r);

    int mid=l+r>>1;

    if (cl<mid) add(lc,l,mid,cl,cr,c);

    if (cr>mid) add(rc,mid,r,ck,cr,c);

    sum[u]=(sum[lc]+sum[rc])%p;

}

ll ask(int u, int l, int r, int cl, int cr){

    if (cl<=l && cr>=r) return sum[u];

    if (tadd[u] || tmul[u]!=1) upd(u,l,r);

    int mid=l+r>>1;

    ll ret=0;

    if (cl<mid) ret+=ask(lc,l,mid,cl,cr);

    if (cr>mid) ret+=ask(rc,mid,r,cl,cr);

    return ret%p;

}

#undef lc

#undef rc

}

namespace NetFlow{

#define INF 0x3f3f3f3f

const int maxn=1003,maxm=10003<<4;

struct Edge{

    int to,nxt,cap,flow,cost;

}ed[maxm];

int head[maxn],ecnt=1,n,m;

void added(int a, int b, int cap){

    ed[++ecnt]=(Edge){b,head[a],cap,0,0};

    head[a]=ecnt;

    ed[++ecnt]=(Edge){a,head[b],0,0,0};

    head[b]=ecnt;

}

queue<int> qu;

int s,t,a[maxn],fr[maxn],fp[maxn];

bool vis[maxn];

//deleted O(n^5)

int MF\_FF(){

    int ans=0;

    while (1){

        memset(vis,0,sizeof(vis));

        memset(a,0,sizeof(a));

        a[s]=INF;

        while (qu.size()) qu.pop();

        qu.push(s);

        vis[s]=1;

        while (qu.size()){

            int u=qu.front(); qu.pop();

            if (u==t) break;

            for (int i=head[u];i;i=ed[i].nxt){

                int v=ed[i].to;

                if (!vis[v] && ed[i].cap>ed[i].flow){

                    vis[v]=1;

                    a[v]=min(a[u],ed[i].cap-ed[i].flow);

                    fp[v]=u;

                    fr[v]=i;

                    qu.push(v);

                }

            }

        }

        if (!a[t]) break;

        ans+=a[t];

        for (int i=t;i!=s;i=fp[i]){

            ed[fr[i]].flow+=a[t];

            ed[fr[i]^1].flow-=a[t];

        }

    }

    return ans;

}

int now[maxn],num[maxn];

int MF\_ISAP(){

    int ans=0,u=s;

    for (int i=1;i<=n;i++) now[i]=head[i],a[i]=1;

    a[t]=0; num[0]=1; num[1]=n-1;

    while (a[s]<n){

        bool ok=0;

        if (u==t){

            int mc=INF;

            for (int i=t;i!=s;i=fp[i])

                mc=min(mc,ed[fr[i]].cap-ed[fr[i]].flow);

            for (int i=t;i!=s;i=fp[i]){

                ed[fr[i]].flow+=mc;

                ed[fr[i]^1].flow-=mc;

            }

            ans+=mc; u=s;

        }

        for (int i=now[u];i;i=ed[i].nxt){

            int v=ed[i].to;

            if (a[u]==a[v]+1 && ed[i].cap>ed[i].flow){

                ok=1;

                fp[v]=u; fr[v]=i;

                now[u]=i; u=v;

                break;

            }

        }

        if (!ok){

            int c=n-1;

            for (int i=head[u];i;i=ed[i].nxt)

                if (ed[i].cap>ed[i].flow) c=min(c,a[ed[i].to]);

            if (--num[a[u]]==0) break; //gap opt

            num[a[u]=c+1]++;

            now[u]=head[u];

            if (u!=s) u=fp[u];

        }

    }

    return ans;

}

int dinic\_dfs(int u, int f){

    int ans=0,w;

    if (u==t) return f;

    for (int i=now[u];i;i=ed[i].nxt){

        int v=ed[i].to;

        if (a[v]==a[u]+1 && (w=dinic\_dfs(v,min(ed[i].cap-ed[i].flow,f)))>0){

            ans+=w;

            ed[i].flow+=w; ed[i^1].flow-=w;

            f-=w; if (f==0) return ans;

            now[u]=i;

        }

    }

    if (!ans) a[u]=-1;

    return ans;

}

int MF\_Dinic(){

    int ans=0;

    while (1){

        memset(vis,0,sizeof(vis));

        memset(a,0,sizeof(a)); //a: level

        qu.clear(); qu.push(s); vis[s]=1;

        while (qu.size()){ //BFS

            int u=qu.fornt(); qu.pop();

            if (u==t) break;

            for (int i=head[u];i;i=ed[i].nxt){

                int v=ed[i].to;

                if (!vis[v] && ed[i].cap>ed[i].flow){

                    qu.push(v);

                    a[v]=a[u]+1;

                    fr[v]=i; fp[v]=u;

                    vis[v]=1;

                }

            }

        }

        if (!vis[t]) break;

        for (int i=1;i<=n;i++) now[i]=head[i];

        ans+=dinic\_dfs(s,INF);

    }

    return ans;

}

int dis[maxn];

int MCMF(){

    int ans=0;

    while (1){

        memeset(vis,0,sizeof(vis));

        memeset(dis,0x3f,sizeof(dis));

        qu.clear(); qu.push(s);

        dis[s]=0; vis[s]=1;

        while (qu.size()){ //spfa

            int u=qu.front(); qu.pop(); vis[u]=0;

            for (int i=head[u];i;i=ed[i].nxt){

                int v=ed[i].to;

                if (ed[i].flow<ed[i].cap && dis[v]>dis[u]+ed[i].cost){

                    dis[v]=dis[u]+ed[i].cost;

                    fr[v]=i; fp[v]=u;

                    if (!vis[v]){

                        vis[v]=1;

                        qu.push(v);

                    }

                }

            }

        }

        if (dis[t]==INF) break;

        int mc=INF;

        for (int i=t;i!=s;i=fp[i]) mc=min(mc,ed[fr[i]].cap-ed[fr[i]].flow);

        for (int i=t;i!=s;i=fp[i]){

            ed[fr[i]].flow+=mc;

            ed[fr[i]^1].flow-=mc;

            ans+=mc\*ed[fr[i]].cost;

        }

    }

    return ans;

}

#undef INF

}

namespace SAM{

const int maxn=100001,alpha=26;

struct Node{

    int l,num; bool vis;

    Node \*p, \*tr[alpha];

    vector<Node \*>ch;

    Node (int \_l):l(\_l){memset(tr,0,sizeof(tr));p=num=vis=0;}

};

Node \*root;

void build(){

    Node \*cur;

    cur=root=new Node(0);

    for (int i=0;i<sl;i++){

        int x=s1[i]-'a';

        Node \*p=cur;

        cur=new Node(i+1);

        for (;p && !p->tr[x];p=p->p)

            p->tr[x]=cur;

        if (!p) cur->p=root;

        else{

            Node \*q=p->tr[x];

            if (p->l+1==q->l) cur->p=q;

            else{

                Node \*r=new Node(-1); r[0]=q[0]; r->l=p->l+1;

                q->p=r; cur->p=r;

                for (;p && p->tr[x]==q;p=p->p) p->tr[x]=r;

            }

        }

    }

}

}

namespace Treap{

const int maxn=100001;

struct Node{

    //x: number, s: sum size of cur and subtree, cnt: cnt of cur num

    Node \*c[2];

    int x,s,r,cnt;

    Node(int \_x){c[0]=c[1]=s=cnt=0;x=\_x;rnd=rand();}

}\_CRT\_MEMCPY\_S\_VALIDATE\_RETURN\_ERRCODE[maxn];

#define lc u->c[0]

#define rc u->c[1]

int trcnt=0;

Node \*open(int x){

    tree[trcnt++]=Node(x);

    return tree+trcnt-1;

}

void upd(Node \*u){

    tr[u].s=tr[lc].s+tr[rc].s+tr[u].ct;

    //more updates...

}

//rt: set lc to root

void rot(Node\* &u, int d){ //0 lt, 1 rt

    Node \*t=u->c[d^1]; u.c[d^1]=t->c[d]; t->c[d]=u;

    t->s=u->s; upd(u); u=t;

}

void ins(Node\* &u, int x){

    if (!u){u=open(x);return;}

    if (x==u->x) {u->cnt++;u->s++; return;}

    int d=x>u->x; u->s++;

    ins(u->c[d],x);

    if (u->c[d]->r>u->r) rot(u,d^1);

}

void delx(Node\* &u, int x){

    if (!u) return;

    if (x==u->x){

        if (tr[p].cnt>1) tr[u].cnt--, tr[u].s--;

        else if (!lc || !rc) u=max(lc+rc);

        else{

            rot(u,lc->r>rc->r);

            u->s--,delx(u->c[x>u->x],x);

        }

    }

    else u->s--,delx(u->c[x>u->x],x);

}

void rank(Node \*u, int x){

    if (!u) return -1;

    if (u.x==x) return lc->s+1;

    if (x>tr[p].val) return lc->s + u->cnt + rank(rc,x);

    else return rank(lc, x);

}

Node\* findx(Node \*u, int x){

    if (!u) return 0;

    if (x==u->v) return u;

    return findx(u->c[x>u->x],x);

}

Node\* findr(Node \*u, int r){

    if (!p) return 0;

    if (x<=lc->s) return findr(lc,r);

    r-=lc->s;

    if (x<=u->cnt) return u;

    r-=u->s;

    return findr(rc,r);

}

#undef lc

#undef rc

}

int main(){

    return 0;

}