**double eps=1e-9;**

**double pi=acos(-1);**

**bool compare(double f,double s)**

**{**

**return (abs(f-s)<eps);**

**}**

**ll add(ll a,ll b,ll mod)**

**{**

**return (a%mod+b%mod)%mod;**

**}**

**ll sub(ll a,ll b,ll mod)**

**{**

**return (a%mod-b%mod+mod)%mod;**

**}**

**ll mul(ll a,ll b,ll mod)**

**{**

**return (a%mod\*b%mod)%mod;**

**}**

**ll sumodd(ll num,ll mod)**

**{**

**return mul(num,num,mod);**

**}**

**ll sumeven(ll num,ll mod)**

**{**

**return mul(num,num+1,mod);**

**}**

**ll sumrange(ll st,ll en,ll num)**

**{**

**return (num\*(st+en)/2);**

**}**

**ll fastpower(ll b,ll p)**

**{**

**ll ans=1;**

**while(p){if (p%2){ans=(ans\*b);}b=b\*b;p/=2;}**

**return ans;**

**}**

**ll fastpowermod(ll b,ll p,ll m)**

**{**

**ll ans=1;**

**while(p)**

**{**

**if (p%2)**

**{**

**ans=((ans%m)\*(b%m))%m;**

**}**

**b=((b%m)\*(b%m))%m;**

**p/=2;**

**}**

**return ans%m;**

**}**

**string makeitbase(ll num,ll base)**

**{**

**string s;**

**while(num)**

**{**

**ll mod=num%base;**

**s+=(mod+'0');**

**num/=base;**

**}**

**reverse(s.begin(),s.end());**

**return s;**

**}**

**bool bit(int num,int i)**

**{**

**return((num>>i)&1);**

**}**

**string makeitstring(ll n)**

**{**

**string ans;**

**bool ch=0;**

**if (n<0){n\*=-1; ch=1;}**

**if (n==0){ans="0";}**

**while (n)**

**{**

**ll mo=n%10;**

**mo+=48;**

**char m=mo;**

**ans=m+ans;**

**n/=10;**

**}**

**if (ch){ans='-'+ans;}**

**return ans;**

**}**

**ll make\_it\_number(string s,ll base)**

**{**

**ll ans=0;**

**for (int i=0;i<s.size();i++)**

**{**

**if (s[i]=='1'){ans+=fastpower(base,i);}**

**}**

**return ans;**

**}**

**ll gcd(ll a,ll b)**

**{**

**while(b!=0)**

**{**

**ll a2=a;**

**a=b;**

**b=a2%b;**

**}**

**return a;**

**}**

**ll lcm(ll a,ll b)**

**{**

**return a/gcd(a,b)\*b;**

**}**

**ll modInverse(ll a,ll m)**

**{**

**int m0 = m;**

**int y = 0, x = 1;**

**if (m == 1)**

**return 0;**

**while (a > 1)**

**{**

**// q is quotient**

**int q = a / m;**

**int t = m;**

**// m is remainder now, process same as**

**// Euclid's algo**

**m = a % m, a = t;**

**t = y;**

**// Update y and x**

**y = x - q \* y;**

**x = t;**

**}**

**// Make x positive**

**if (x < 0)**

**x += m0;**

**return x;**

**}**

**void initfact()**

**{**

**fact[0]=1;**

**for (int i=1;i<N;i++)**

**{**

**fact[i]=fact[i-1]\*i;**

**fact[i]%=M;**

**}**

**for (int i=0;i<N;i++)**

**{**

**invfact[i]=modInverse(fact[i],M)%M;**

**}**

**}**

**ll npr(ll n,ll r)**

**{**

**return (fact[n]\*invfact[n-r])%M;**

**}**

**/\***

**ll modth[3];**

**ll cntdivrange(ll l,ll r,ll num){return (r/num)-(l/num)+(l%num==0);}**

**for (int i=0;i<3;i++)**

**{**

**// cnt how many number in range mod 3=0 mod 3 =1 mod 3 =2**

**modth[i]=cntdivrange(l+i,r+i,3);**

**}**

**\*/**

**ll count\_factorial\_last\_zeros(ll fact,ll primefactor,ll how\_many\_prime\_factor\_in\_this\_base)**

**{**

**ll ans=0;**

**while(fact/primefactor>0)**

**{**

**ans+=(fact/=primefactor);**

**}**

**return (ans/how\_many\_prime\_factor\_in\_this\_base);**

**}**