vector<int>prime;

bool status[MX+2];

void **PrimeGenerate**(int n)

{

int sq=**sqrt**(n);

**for**(int i=3; i<=sq; i+=2)

{

**if**(!status[i])

**for**(int j=i\*i; j<=n; j+=(i<<1))

status[j]=**true**;

}

status[0]=status[1]=**true**;

**for**(int i=4;i<=n;i+=2) status[i]=**true**;

prime.**push\_back**(2);

**for**(int i=3; i<=n; i+=2)

**if**(!status[i])

prime.**push\_back**(i);

**return**;

}

vector<int> PrimeFactorize(int n)

{

vector<int> primeDiv\_pow;

int pos=0;

while(n>=prime[pos] && pos<pl)

{

int cnt=0;

while(!(n%prime[pos]))

{

n/=prime[pos];

cnt++;

}

if(cnt)

{

primeDiv\_pow.push\_back(prime[pos]);

primeDiv\_pow.push\_back(cnt);

}

pos++;

}

if(n!=1)

{

primeDiv\_pow.push\_back(n);

primeDiv\_pow.push\_back(1);

}

return primeDiv\_pow;

}

int numberOfDivisor(int n)

{

vector<int>primeDiv\_pow=PrimeFactorize(n);

int vl=primeDiv\_pow.size(),numOfDiv=1;

for(int i=1;i<vl;i+=2)

numOfDiv\*=primeDiv\_pow[i]+1;

return numOfDiv;

}

int sumOfDivisor(int n)

{

vector<int>primeDiv\_pow=PrimeFactorize(n);

int vl=primeDiv\_pow.size(),sumOfDiv=1;

for(int i=1;i<vl;i+=2)

sumOfDiv\*=(pow(primeDiv\_pow[i-1], primeDiv\_pow[i]+1)-1)/(primeDiv\_pow[i-1]-1);

return sumOfDiv;

}

int CoPrime(int n)

{

vector<int>primeDiv\_pow=PrimeFactorize(n);

int vl=primeDiv\_pow.size();

for(int i=0;i<vl;i+=2)

{

n/=primeDiv\_pow[i];

n\*=(primeDiv\_pow[i]-1);

}

return n;

}