// 素数筛 欧拉筛 欧拉函数

bool isprime[maxn];

int prime[maxn];

int sum[maxn];

int pcnt;

void init() {

memset(isprime, true, sizeof(isprime));

memset(prime, 0, sizeof(prime));

pcnt = 0;

}

void getPrime() {

init();

prime[0] = prime[1] = 0;

for(int i = 2; i <= maxn; i++) {

if(isprime[i]) prime[++pcnt] = i;

for(int j = 1; j <= pcnt; j++) {

if(i \* prime[j] > maxn) break;

isprime[i\*prime[j]] = 0;

if(i % prime[j] == 0) break;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*

bool isprime[maxn];

int prime[maxn], cnt = 0;

void prim()

{

mem(isprime);

for(int i = 2; i <= maxn; i++)

if( isprime[i] )

{

prime[cnt++] = i;

for(int j = i\*2; j <maxn; j += i)

isprime[i] = 0;

}

}

void foo()

{

mem(isprime);

for(int i = 2; i < maxn; i++)

{

if( isprime[i])

prime[cnt++] = i;//cnt返回小于n的素数的个数

for(int j = 0; j < cnt; j++) //j < cnt

{

if( i\*prime[j] >= maxn ) break;

isprime[ i\*prime[j] ] = false;//找到的素数的倍数不访问

if( i%prime[j] == 0 ) break;//关键

//prime数组 中的素数是递增的,当 i 能整除 prime[j]，那么 i\*prime[j+1]

//这个合数肯定被 prime[j] 乘以某个数筛掉

//prime[j]必定是prime[j]\*i的最小因子

}

}

}

/\*线性筛O(n)时间复杂度内筛出maxn内欧拉函数值\*/

int m[maxn],phi[maxn],p[maxn],pt;//m[i]是i的最小素因数，p是素数，pt是素数个数

//求phi欧拉函数

void getphi()

{

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

phi[i]=i;

for(int i=2;i<N;i++)

if(phi[i]==i)

for(int j=i;j<N;j+=i)

phi[j]=phi[j]-phi[j]/i;

}

int make()

{

phi[1]=1;

int N=maxn;

int k;

for(int i=2;i<N;i++)

{

if(!m[i])//i是素数

p[pt++]=m[i]=i,phi[i]=i-1;

for(int j=0;j<pt&&(k=p[j]\*i)<N;j++)

{

m[k]=p[j];

if(m[i]==p[j])//为了保证以后的数不被再筛，要break

{

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

/\*这里的phi[k]与phi[i]后面的∏(p[i]-1)/p[i]都一样（m[i]==p[j]）只差一个p[j]，就可以保证∏(p[i]-1)/p[i]前面也一样了\*/

break;

}

else

phi[k]=phi[i]\*(p[j]-1);//积性函数性质，f(i\*k)=f(i)\*f(k)

}

}

}

int main()

{

foo();

make();

rep(i,1,50)

cout << phi[i] << ' ';

}