Number Theory - 4

Q.1)- You will be given a number N, find the sum of its all prime divisors. (N<=10^6)

12-> 1,2,3,4,6,12
Sum of all prime divisors = (2+3)=5;

Find its all of its divisors and then check how many of them are prime number.

O(Nlog(log(N))) + O(N) => O(Nlog(log(N)))

Q.2)- Now you have to solve the same above problem but for q number of queries. (q<=10^6)

Time Complexity of Native solution-> O(q* Nlog(log(N)))

```
#include <bits/stdc++.h>
using namespace std;

int main(){

        const int MAX=1000000;
        bool is_prime[MAX+1];
        int Sum[MAX+1];
        // Sum[i]-> sum of its all prime divisors.
        memset(is_prime,true,sizeof(is_prime));
```

```cpp
        is_prime[0]=is_prime[1]=false;

        for(int i=2;i<=MAX;i++){
            if(is_prime[i]==true){
                for(int j=i;j<=MAX;j+=i){
                    if(j>i) is_prime[j]=false;
                    sum[j]+=i;
                }
            }
        }
        //O(NlogN)
        i=2(Prime);
        2,4,6,8,10....
        Maked all these numbers as Non prime number
        and Sum[j]+=i;
        int q;
        cin>>q;
        while(q--){ //O(q)
            int n;
            cin>>n;
            cout<<Sum[n]<<endl;
        }

        //OverAll timeComplexity = O(N(logN))
        return 0;
}

//log(sqrt(N)) = ½ log(N)
```

Q.3)- You will be given a number N, find the number of its divisors. (N<=10^6)

```
Int count=0;
for(int i=1;i*i<=N;i++){
        if(N%i==0){
                Int first_divisor = i;
                Int second_divisor = (N/i);
                if(first_divisor!=second_divisor) count+=2;
                Else count++;
        }
}
cout<<count<<endl;
Time Complexity -> O(sqrt(N))
```

Q.4)- Now you have to solve the same above problem but for q number of queries. (q<=10^6)

```
Brute Force Time Complxity -> O(q*sqrt(N)) == 10^9

Points->
        N-> z1^k1 * z2^k2 * z3^k3 * …….
        zi-> prime number.
        Number of divisors = (k1+1)*(k2+1)*(k3+1)......

Bool is_prime[MAX+1];
Int SPF[MAX+1];
```

```
Bool is_prime[MAX+1];
Int SPF[MAX+1];
// NLog(logN))
Int q;
cin>>q;
while(q--){
      Int n;
      cin>>n;
      Int ans=1; (12)
      while(n>1){
            Int k=0;
            Int spf = SPF[n];
            while(n%spf==0){
                  n/=spf;
                  k++;
            }
            ans=ans*(k+1);
      }
      cout<<ans<<endl;
}
Time complxity -> O(qlogn);

N = 2^20
Spf = SPF[N] = 2;
while(N%spf==0) n/=2;
It will run 20;
Log2(N);

N= 2^10 * 3^10
```

Q.

Sol:-

Number of packages*Number of shovels in that package = n

Number of packages = n/(number of shovels in that package)

->Number of shovels in that package must divide n.

-> number of shovels is a divisor of n.

-> number of shovels<=k

→ the max divisor of n <=k. -> ans

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
int32_t main()
{
    int t;
    cin>>t;
    while(t--){ //for(int i=0;i<t;i++)
        int n,k;
        cin>>n>>k;
        vector<int> divisors;
        for(int i=1;i*i<=n;i++){
            if(n%i==0){
                divisors.push_back(i);
                divisors.push_back(n/i);
            }
        }
        int ans=n;
        for(int i=0;i<divisors.size();i++){
            if(divisors[i]<=k){
```

```
            ans = min(ans,n/divisors[i]);
        }
    }
    cout<<ans<<endl;
  }
}
```

Q: https://codeforces.com/problemset/problem/776/B

n=3 -> jewelery -> 1,2,3
        Price   -> 2,3,4
    2 colors ->   1,1,2
                   1,2,2
n=7
Prices -> 2,3,4,5,6,7,8
       -> 1,1,2,1,2,1,2
n=2 -> 2,3
n=1,2 -> number of color = 1
Else number of color =2
n=12 -> 2,3,4,5,6,7,8,9,10,11,12,13
      -> 1,1,2,1,2,1,2,2,2,1,2,1

Q.

Eg 20 8
Divisors of 20 -> 1,2,4,5,10,20
Divisors of 8 -> 1,2,4,8
10 2 8 1 2 4 1 20 4 5
20 -> 1,2,4,5,10,20

Euler Totient Function:-

phi(n) = count of numbers from 1 to n that are coprime with n.
Coprime:- 2 numbers x and y are coprime if gcd(x,y)=1.
phi (2)=1 (1)
phi(4)=2   (1,3)
phi(8)=4 (1,3,5,7)
Number p -> prime number
phi(p) -> 1....p-1 = p-1
phi(p^2) -> p^2-p
$9 \to 3^2 \to 3,6,9 = 9/3 = p^2/p$
$phi(p^k) \to p^k \to p, 2*p, 3*p, 4*p...p^k \to p^{(k-1)}$
$p^k = p+(n-1)*p$
$p^{(k-1)} = 1+n-1$
$N = p^{(k-1)}$
phi(p^k) = p^k-p^(k-1)
N = (p1^k1).(p2^k2)....
Where p1,p2,... are prime numbers.
phi(a*b) = phi(a).phi(b) if a and b are coprime.
$phi(n) = phi(p1^{k1} . p2^{k2} ....)$
$Phi(n) = phi(p1^{k1}).phi(p2^{k2}).....$

phi(n) = (p1^k1-p1^(k1-1))*(p2^k2-p2^(k2-1))....

= {p1^k1.p2^k2...}(1-1/p1)(1-1/p2).....

= n*(1-1/p1)*(1-1/p2)....

= n *pi{1-1/pi}

```
Int phi[1000001];
for(int i=0;i<=1e6;i++){
      phi[i]=i;
}
for(int i=2;i<=1e6;i++){
      if(phi[i]==i){
            for(int j=i;j<=1e6;j+=i){
                  Phi[j] = phi[j]-phi[j]/i;
            }
      }
}
```