Q. Given a number n, find out whether it is prime or not.

```cpp
bool isPrime = true;
for(int i=2; i*i <= n; i++){
    if(n % i == 0){
        isPrime = false;
        break;
    }
}
```

Q. Given a number n, list out its divisors.

```cpp
vector<int> divisors;
for(int i=1; i*i <= n; i++){
    if (n%i == 0){
        divisors.push_back(i);
        if(i*i != n){
            divisors.push_back(n/i);
        }
    }
}
```

Q. Given a number n, find out its prime factorization.

```cpp
vector<pair<int,int>> factorization;
for(int i=2; i*i <= n; i++){
    int count = 0;
    while(n % i == 0){
        count++;
        n /= i;
    }
    if(count > 0){
        factorization.push_back({i, count});
    }
}
if(n != 1){
    factorization.push_back({n, 1});
}
```

Q. [Almost All Divisors]

Sort the divisors
Assume min*max = required number = x. Find out all divisors of x.
Tally with the given list of divisors.

```
int n;
cin >> n;
vector<long long> v(n);
for (int i = 0; i < n; i++)
    cin >> v[i];
sort(v.begin(), v.end());
long long prod = v[0] * v[v.size() - 1];
vector<long long> fac;
for (long long i = 2; i * i <= prod; i++){
    if (prod % i == 0){
        fac.push_back(i);
        if (prod != i * i)
            fac.push_back(prod / i);
    }
}
sort(fac.begin(), fac.end());
bool doable = true;
if (fac.size() != n)
    cout << "-1\n";
else{
    for (int i = 0; i < n; i++)
        if (v[i] != fac[i]){
            doable = false;
            break;
        }
    cout << (doable ? prod : -1) << "\n";
}
```

Sieve of Eratosthenes:

```
const int lim = 1e7+5;
vector<bool> isPrime(lim, true);
isPrime[0] = isPrime[1] = false;
for(int i=2; i*i < lim; i++){
    if(isPrime[i]){
        for(int j = i*i; j<lim; j+=i)
            isPrime[j] = false;
    }
}
```

Smallest Prime Factor:

```
const int lim = 1e7 + 5;
vector<int> spf(lim);
for(int i=0; i<lim; i++)
    spf[i] = i;
for (int i = 2; i * i < lim; i++){
    if (spf[i]==i){
        for (int j = i * i; j < lim; j += i)
            if(spf[j] == j)
                spf[j] = i;
    }
}
```

Largest Prime Factor:
(Note that i*i < lim and starting inner loop with j = i*i won't work here)

```
const int lim = 1e7 + 5;
vector<int> lpf(lim);
for(int i=0; i<lim; i++)
    lpf[i] = i;
for (int i = 2; i < lim; i++){
    if (lpf[i]==i){
        for (int j = i; j < lim; j += i)
            lpf[j] = i;
    }
}
```

Segmented Sieve

Print all primes between given range [L,R]; 1<=L,R<=1e7

```
vector<bool> isPrime(R+1, true);
isPrime[0] = isPrime[1] = false;
for(int i=2; i*i <= R; i++){
    if(isPrime[i]){
        for(int j = i*i; j<lim; j+=i)
            isPrime[j] = false;
    }
}
for(int i=L;i<=R;i++)
{
     if(isPrime[i])
     cout<<i<<" ";
}
```

What if Range of L,R changes to approx 1e12?
Consider 1<=L,R<=1e12, R-L<=1e5
We will apply the sieve on a segment. Here comes the concept of segmented sieve

-> **We need to have primes only upto square root of R because using them we can find whether a particular number in the range is prime or not.**

Proper Solution

```
#define int long long
void solve()
{
        int L,R;
        cin>>L>>R;
        int sqR = sqrt(R);
        vector <int> prime(sqR+1,1);
        vector <int> store; // stores prime upto sqrt(R)

        // Storing Prime Number upto sqrt(R) using normal sieve
        for(int i=2;i<=sqR;i++)
        {
                if(!prime[i])
                continue;
                store.push_back(i);
                for(int j=i*i;j<=sqR;j+=i)
                prime[j] = 0;
        }
```

//Idea is to mark number as prime in the whole range from 1...r mark values in the range [L,R]

```
	vector <int> isPrime(R-L+1,1);
	//Marking number in the Range [L,R] as prime or not using indexes [0,1,2....R-L]
	for(auto it:store)
	{
		int start = it*it;
		start = max(start,((L+it-1)/it)*it); // Find first number
		for(int j=start;j<=R;j+=it)
		isPrime[j-L]=0;
	}
	if(L==1)
	isPrime[0]=0;
	for(int i=L;i<=R;i++)
	{
		if(isPrime[i-L])
		cout<<i<<"\n";
	}
	cout<<"\n";

}
```

Time Complexity -> (R-L+1)*loglog(R) + Sqrt(R)loglog(sqrt(R))

Class Discussion :

```
//Q. Print all prime numbers in the range [L,R]   L,R <= 1e12    R-L <= 1e5

int n = sqrt(R); // storing prime upto sqrt(R) is sufficient to check primality upto R
vector <bool> isPrime(n+1,true);
isPrime[0] = isPrime[1] = false;
vector <int> primes;

//Normal Seive upto sqrt(R)
for(int i=2;i*i<=n;i++)
{
    if(isPrime[i])
    {
        for(int j=i*i;j<=n;j+=i)
        isPrime[j] = false;
    }
}

// Storing primes upto sqrt(R)
for(int i=2;i<=n;i++)
if(isPrime[i])
primes.push_back(i);

vector <bool> checkPrime(R-L+1,true); //number L,L+1,L+2....R -> indexes 0,1,2.....R-L
```

```
`
//[L,R] -> [399,599]

//2 ->4,6,8,10,12....R  O(R) -> not acceptable
//2->400,402,404....,598 O(R-L) -> acceptable
//[L,R] -> first number which is a multiple of prime 'pr'

//(ceil(L/pr))*pr

// L = k*pr + c  // if c==0  first number = k*pr else (k+1)*pr
// ceil(L/pr) = k if(c==0)  k*pr
// ceil(L/pr) = k+1 if(c!=0) (k+1)*pr

// ceil(a/b) -> (a+b-1)/b


for(auto pr:primes)
{
    int start = ((L+pr-1)/pr)*pr;
    start = max(start,pr*pr);
    for(int j = start ;j<=R;j+=pr)
    checkPrime[j-L] = false;
}
for(int i=0;i<=R-L;i++)
if(checkPrime[i])
ans.push_back(i+L);


Proof -> Why (a+b-1)/b  gives ceil(a/b)
a = kb
->k

(a+b-1)/b

x = (kb + b-1)   ->k    kb <x <(k+1)b    x/b -> k

a  = kb+c
->k+1


x = (kb+c +b - 1)  (k+1)/b < x
x/b -> (k+1)
```

Refer -> https://cp-algorithms.com/algebra/sieve-of-eratosthenes.html#toc-tgt-7

Solve -> https://www.spoj.com/problems/PRIME1/……….