L15 : Number Theory 1

1-Tut : Find Prime Numbers From 1 to N

Send Feedback

Given a number N, find number of primes in the range [1,N].

Input Format:

The only line of input consists of a number N

Output Format:

Print the number of primes in the range [1,N].

Constraints:

1 ≤ N ≤ 10^6

Sample Input :

3

Sample Output -

2

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1000005;
bool sieve[N];

int main()
{
    for(int i = 0 ; i < N ; ++i) {
        sieve[i] = true;
    }
    sieve[0] = sieve[1] = false;

    for(int i = 2 ; i*i <= N ; ++i) {
        if(sieve[i]) {
            for(int j = i*i ; j < N ; j += i) {
                sieve[j] = false;
            }
        }
    }

    int n; cin >> n;

    int count = 0;
    for(int i = 0 ; i <= n ; ++i) {
        if(sieve[i]) ++count;
    }

    cout << count << '\n';

    return 0;
}
```

2-Tut : GCD

Send Feedback

Calculate and return GCD of two given numbers x and y. Numbers are within range of Integer.

Input format :

First line of Input will contain T(number of test cases), each test case follows as.
x and y (separated by space)

Output format :

Print GCD of x and y for each test case in newline

Constraints:

1 <= T <= 10^5
1 <= x, y <= 10^9

Sample Input 1:

```
1
20 5
```

Sample Output 1:

```
5
```

Sample Input 2:

```
1
96 14
```

Sample Output 2:

```
2
```

```
#include<bits/stdc++.h>
using namespace std;
int gcd(int x,int y){
    if((x == 0) || y == 0 ){
        return x^y;
    }
    if(x > y){
        return gcd(y,x%y);
    }

    return gcd(x,y%x);
}
int main(){

    // write your code here
    int t;cin>>t;
    while(t--){
        int x,y; cin>>x>>y;
        cout<<gcd(x,y)<<endl;
    }
    return 0;
}
```

3-Ass : Super Prime

Send Feedback

A number is called super-prime if it has exactly two distinct prime divisors

Example 10 , 6

You are supposed to find the count of super-prime numbers between 1 and N (inclusive).

Input Format:

Contain an integer N

Output Format:

Print the number of super prime between [1, N]

Constraints:

1 <= N <= 10^6

Sample Input 1:

10

Sample Output 1:

2

Sample Input 2:

25

Sample Output 2:

10

Explanation:

The super-primes are: 6, 10, 12, 14, 15, 18, 20, 21, 22, 24.

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1000005;
long long sieve[N];

int main()
{
    for(int i = 0 ; i < N ; ++i) {
        sieve[i] = 0;
    }

    for(int i = 2 ; i < N ; ++i) {
        if(!sieve[i]) {
            for(int j = 2*i ; j < N ; j += i) {
                ++sieve[j];
            }
        }
    }

    int n; cin >> n;

    int count = 0;
    for(int i = 0 ; i <= n ; ++i) {
        if(sieve[i] == 2) ++count;
    }

    cout << count << '\n';

    return 0;
}
```

4-Ass : Ninja and Flowers

Send Feedback

Ninja wants to get N flowers and he will pay i + 1 amount of money for the Ith flower, example (if n=3 he will pay {2,3,4})

Now he wants to pack these N flowers in boxes of different colours. With one condition if the cost of a flower is a prime divisor of another flower they needed to be of a different colour.

As we know that ninja is a little money minded he wants to minimize the number of different colours of boxes that he needs.

Input Format:

The only line of input will contain an integer N (number of flowers).

Output Format:

In first-line print K, the minimum number of different colour boxes that are needed to pack the flowers. Next line contains K space-separated integers in sorted order denoting the counts of the different coloured boxes.

Constraints:

1 <= N <= 2*10^5

Sample Input:

```
4
```

Sample Output:

```
2

1 3
```

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

```
#define int long long
#define double long double

const int N = (int) 1e6+5;
vector<bool> sieve;

int32_t main()
{
        ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);

        sieve = vector<bool>(N, true);
        for(int i = 2 ; i*i <= N ; ++i) {
                if(sieve[i]) {
                        for(int j = i*i ; j < N ; j += i) {
                                sieve[j] = false;
                        }
                }
        }

        int n; cin >> n;

        if(n <= 1) {
                cout << 1 << '\n';
        }
        else {
                cout << 2 << '\n';
        }

        int cp = 0, cnp = 0;
        for(int i = 2 ; i <= n+1 ; ++i) {
                if(sieve[i]) ++cp;
                else ++cnp;
        }

        cout << min(cp, cnp) << ' ' << max(cp, cnp) << '\n';

        return 0;
}
```

5-Ass : Special Prime

Send Feedback

Special Prime is a prime number that can be written as the sum of two neighbouring primes and 1.

You are given an integer N and you are supposed to find the number special prime in the range: [1, N].

Example of special prime 19 = 7 + 11 + 1

Neighbouring primes are prime number such that there is no other prime number between them.

Input Format:

An integer N.

Output Format:

Print the number of special primes

Constraints:

1 <= N <= 2*10^5

Sample Input:

27

Sample Output:

2

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
#define double long double

const int N = (int) 1e6+5;
vector<bool> sieve;

int32_t main()
{
        ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);

        int n; cin >> n;

        sieve = vector<bool>(N, true);
        for(int i = 2 ; i*i <= N ; ++i) {
                if(sieve[i]) {
                        for(int j = i*i ; j < N ; j += i) {
                                sieve[j] = false;
                        }
                }
        }

        vector<int> primes;
        for(int i = 2 ; i < N ; ++i) {
                if(sieve[i]) primes.emplace_back(i);
        }

        unordered_set<int> set;
        for(int i = 1 ; i < primes.size() ; ++i) {
                set.insert(primes[i-1] + primes[i]);
```

```
    }

    int count = 0;
    for(int i = 2 ; i <= n ; ++i) {
        if(sieve[i] && set.count(i-1)) ++count;
    }

    cout << count << '\n';

    return 0;
}
```