→ Sieve Algorithm : To find Prime Numbers Optimally.

→ Suppose we have $q = 10^7$ queries and $n = 10^7$ numbers to check prime in each queries.

So, if we use Square root method (which was quite fast), still we will have a Time Complexity of $O(q) * O(sqrt(n))$.

∴ This will give TLE.

-> So, To solve this problem, we have to compute all prime numbers using Sieve Algorithm.

```
CONST INT N = 1e7+10;
```

-> Initializing a vector with N size, initially with all true values.

```
VECTOR<BOOL> ISPRIME(N, 1);
```

-> Sieve Algorithm

-> Suppose we have n = 30
Write down numbers till 30.

~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~
11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~
~~21~~ ~~22~~ 23 ~~24~~ ~~25~~ ~~26~~ ~~27~~ ~~28~~ 29 ~~30~~

STEPS:

0. Initially we consider all the numbers till n as prime numbers, and at the end:
- CROSSED = Non Prime
- UNCROSSED = Prime

1. 1 is non-prime (we all know). So, simply cross it and move to next number.

2. The next number is 2, cross all the multiples of 2 till 30.

3. Now move to next uncrossed number which is 3, do same with 3, cross all its multiples.

(ignore already crossed..)

4. Same process again, we get 5 uncrossed.

5. Follow this process until we consider all the uncrossed numbers.

6. Finally, we are left with:
UNCROSSED = 2, 3, 5, 7, 11, 13, 17, 19, 23, 29
These are all prime numbers till 30.

NOTE: Whenever we move to next uncrossed number, that number is always gonna be a prime number.

```
INT MAIN()
{
    -> Sieve Algorithm, Time Complexity
       O(N * log(log(N)))

    -> Setting 0 and 1 as non prime (as
       we all know that.
    ISPRIME[0] = ISPRIME[1] = FALSE;
    -> Pre computing all the primes till N.
    FOR (INT i=2; i<N; i++)
    {
        -> If i is prime
        IF (ISPRIME[i] == TRUE)
        {
            FOR (INT j=2*i; j<N; j+=i)
            {
```

-> changing all multiples of i to False.

```
                IsPRIME[j] = FALSE;
            }
        }
    }
```

-> Queries

```
INT Q;
CIN >> Q;

WHILE(Q--)
{
    INT n;
    CIN >> n;

    IF (IsPRIME[n])
    {
        COUT << "PRIME" << "\n";
    }
    ELSE
    {
        COUT << "NON PRIME" << "\n";
    }
}
```

-> Time Complexity = O(q) * O(1)

```
RETURN 0;
}
```