segmented sieve PRIME1 - Prime Generator SPOJ #numberTheory

```
/*
    --> Segmented Sieve
    This algorithm helps you find out all the prime numbers
    with in a given range [a, b]
    in this case the max value of b can be upto 10^9;
    let's assume, sqrt(10^9) = 32000
    therefore, MAX = 32000, you'll have to change vale of MAX
    depending upon the value of b.
*/
#include <limits.h>
using ll = long long;
#define MAX 32000

vector <int>* sieve() {
    bool isPrime[MAX];
    for (int i = 0; i < MAX; ++i) isPrime[i] = true;
    for (int i = 3; i * i <= MAX; i += 2) {
        if (isPrime[i]) {
            for (int j = i * i; j <= MAX; j += i) {
                isPrime[j] = false;
            }
        }
    }
    vector <int>* primes = new vector<int> ();
    primes->push_back(2);
    for (int i = 3; i < MAX; i += 2) {
        if (isPrime[i]) {
            primes->push_back(i);
        }
    }
    return primes;
}
void printPrimes(ll l, ll r, vector<int>* & primes) {
    bool isPrime[r-l+1];
    for (int i = 0; i < r-l+1; ++i) isPrime[i] = true;
    if (l == 1) isPrime[0] = false;
    for (int i = 0; primes->at(i)*(ll) primes->at(i) <= r; ++i) {
        int currPrime = primes->at(i);
        ll base = (l/currPrime)*currPrime;
        if (base < l) base += currPrime;
        for (ll j = base; j <= r; j += currPrime) {
            isPrime[j-l] = false;
        }
        if (base == currPrime) isPrime[base - l] = true;
    }
    for (int i = 0; i < r - l + 1; ++i) {
        if (isPrime[i]) cout << (i+l) << endl;
    }
    puts("");
}
int main() {
    vector <int>* primes = sieve();
    int t;
    cin >> t;
    while (t--) {
        ll l, r;
        cin >> l >> r;
        printPrimes(l, r, primes);
    }
}
```