sieve uisng bitest #numbertheory

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

const int mx = 10000;
bitset<mx> bs;
vector<int> primes;
void sieve(long long upper_bound) {
    bs.set();
    bs[0] = bs[1] = 0;
```

bs.set(): sets all of the bits in the *bs* to 1 *bs.reset*(): resets all of the bits in *bs* or sets to 0

```
    primes.push_back(2);
    for(long long i = 3; i <= upper_bound + 1; i += 2) {
        if(bs[i]) {
            for(long long j = i * i; j <= upper_bound + 1; j += 2*i)
                bs[j] = 0;
            primes.push_back((int) i);
        }
    }
}
void isPrime (int n) {
    if (n <= 1) return false;
    if (n == 2) return true;
    if (n % 2 != 0 && bs[n]) return true;
}
```

Always use the *isPrime*() function to find out if a number is prime or not. Use the *Primes* vector to get all the primes upto the given *upper_bound*.

```
int main() {
    sieve(1000);
    return 0;
}
```

To get the 1000*th* prime number print out the value of *primes*[999]

```
    cout << primes[1000-1] << endl;
```