PRIME CHECK:

```
bool prime (int n) {
    if (n < 2) return false;
    if (n <= 3) return true;
    if (n % 2 == 0) return false;

    for ( i = 3; i <= sqrt(n); i++) {
        if (n % 1 == 0) {
            return false;
        }
    return true;
}
```

SIEVE :

```
void sieve (int n) {
    int prime[n+3];
    memset(prime, 0, sizeof(prime));

    for (int i=2; i <= sqrt(n); i++) {
        if (prime[i] == 0) {
            for (int j = i*i; j <= n; j += i) {
                prime[j] = 1;
            }
        }
    }

    for (int i = 2; i <=n; i++) {
        if (prime[i] == 0) {
            cout << i << " ";
        }
    }
    cout << endl;
}
```

Prime fact:

```
void primefact (int n) {
    bool first = true;

    for (int i = 2; i <= sqrt(n); i++) {
        if (n % i == 0) {
            int cnt = 0;

            while (n % i == 0) {
                cnt++;
                n = n / i;
            }cout << i << "^" << cnt << ",";
        }
    }
    if (n > 1) {
        cout << n << "^" << 1 << endl;
    } else {
        cout << endl;
    }
}
```

Binary expo:

```
ll pow(ll base,ll n,ll mod){
 ll result=1;
  while(n){
    if(n%2==1){

result=(result*base)%mod;
      n--;
    }else{
      base=(base*base)%mod;
      n/=2;
    }
  }
  return result;
}
int main(){
 ll t; cin>>t;
  while(t--){
    ll n,m; cin>>n>>m;

cout<<pow(n,m,1e9)<<endl;
  }
  return 0;
}
```

NOD:

```
int main () {
  int tc;
  cin >> tc;
  ll arr[1000005] = {};

  for(int i=1; i<1000005; i++){
    for(int j=i; j<1000005; j+=i){
      arr[j]++;
    }
  }

  while (tc--) {
    int x;
    cin >> x;
    cout << arr[x] << endl;
  }
  return 0;
}
```

Kth prime:

```
const long long x=90000001;
vector<bool>vec(x,true);
vector<int> ans;

void sieve(){
  for (int i=2;i<=x;i++){
    if (vec[i]==true){
      ans.push_back(i);
      for (int j=i*i;j<=x;j+=i){
        vec[j] = false;
      }
    }
  }
}
int32_t main(){
    int q; cin>>q;
    sieve();
    while (q--){
        int n; cin >> n;
        cout<<ans[n-1]<<endl;
    }
    return 0;
}
```