**Sum of all prime numbers between 1 and N.**

[maths](http://www.practice.geeksforgeeks.org/tag-page.php?tag=maths&isCmp=0)

Given a positive integer N, calculate the sum of all prime numbers between 1 and N.  
  
**Input:**  
The first line of input contains an integer T denoting the number of test cases.  
The first line of each test case is N.  
  
**Output:**  
Print the sum of all prime numbers between 1 and N.  
  
**Constraints:**  
1 ≤ T ≤ 50  
1 ≤ N ≤ 1000000  
  
**Example:**  
**Input:**  
2  
5  
10  
  
**Output:**  
10  
17

\*\*For More Examples Use Expected Output\*\*

<http://www.practice.geeksforgeeks.org/problem-page.php?pid=651>

#include <iostream>

#include <stdio.h>

#include <math.h>

#include <vector>

#define ll long long int

using namespace std;

std::vector<ll> SieveOfEratosthenes(ll n)

{

    std::vector<ll> res;

    // Create a boolean array "prime[0..n]" and initialize

    // all entries it as true. A value in prime[i] will

    // finally be false if i is Not a prime, else true.

    bool prime[n+1];

    memset(prime, true, sizeof(prime));

    for (ll p=2; p\*p<=n; p++)

    {

        // If prime[p] is not changed, then it is a prime

        if (prime[p] == true)

        {

            // Update all multiples of p

            for (ll i=p\*2; i<=n; i += p)

                prime[i] = false;

        }

    }

    // Print all prime numbers

    ll sum =0;

    res.push\_back(0);

    for (ll p=2; p<=n; p++) {

       if (prime[p]) {

         // cout << p << " ";

          sum += p;

        }

        res.push\_back(sum);

     }

    return res;

}

int main() {

    int t;

    scanf("%d", &t);

///precomputo las sumas de 1 a N antes de cada caso de prueba

    std::vector<ll> sumas =  SieveOfEratosthenes(1000000);

*/\*std::vector<ll> sumas =  SieveOfEratosthenes(20);*

*for(int i =0; i<sumas.size(); i++) {*

*cout << sumas[i] << " ";*

*}\*/*

    while(t--) {

      ll n;

      scanf("%lld", &n);

//luego computo cada caso de prueba en O(1) tiempo

      cout << sumas[n-1] << endl;

    }

   // system("pause");

    return 0;

}