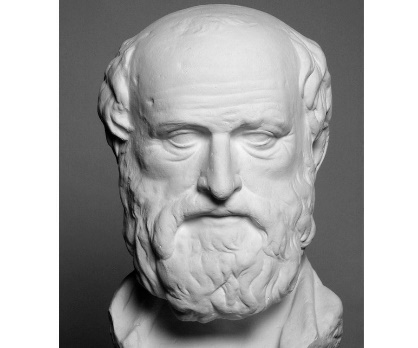
**Lab practice Problems 1**

|  |
| --- |
| 1. Write a function isPrimeCount that takes an integer **n** as argument and checks whether it is prime or not using count approach (chech how many divisors **n** has between 1 to **n**) |
| 1. Write a function isPrimeFlag that takes an integer as argument and checks whether it is prime or not using flag approach (check whether **n** has any divisor between 2 to **n-1)** |
| 1. Write a program that counts the number of primes between 1 to N using isPrimeCount when N equals to 2. 105 3. 106 4. 107   In each case, record the time for execution in seconds using clock. (Check next page to see how to measure runtime in C). |
| 1. Write a program that counts the number of primes between 1 to N using isPrimeFlag when N equals to 2. 105 3. 106 4. 107   In each case, record the time for execution in seconds using clock. |
| 1. Modify isPrimeCount and isPrimeFlag by running the loop between up to **n/2.**   Now, repeat step 3 and 4 again. |
| 1. Modify isPrimeCount and isPrimeFlag by running the loop between up to **√n.**   Now, repeat step 3 and 4 again. |
| 1. Write a program that counts the number of primes between 1 to N using **Sieve of** **Eratosthenes** *(see next page***)** when N equals to 2. 105 3. 106 4. 107   In each case, record the time for execution in seconds using clock. |
| 1. Can you come up with a method that works even faster than Sieve of Eratosthenes? Try yourself by searching. [Bonus] |

**Sieve of Eratosthenes**

* Create an array of consecutive integers from 2 to n: (2, 3, 4, …, n).
* Initially, let p equal 2, the first prime number.
* Starting from p2, count up in increments of p and mark each of these numbers in the array. These numbers will be p(p+1), p(p+2), p(p+3), etc.
* Find the first number greater than p in the list that is not marked. If there was no such number, stop. Otherwise, let p now equal this number (which is the next prime), and repeat from step 3.
* The numbers that will be left unmarked are primes.

**Measuring runtime in C:**

clock\_t start = clock();

//The program

clock\_t end = clock();

float seconds = (float)(end - start) / CLOCKS\_PER\_SEC;

printf("Time taken: %.5f",seconds);