# Assignment 3

CS2340

The Sieve of Eratosthenes

There are several ways to find prime numbers. A standard one is to divide the number x by the numbers 2…sqrt(x) and if it divides evenly by any of them, it is prime. However, you are to write rather different algorithm, discovered by the ancient Greek polymath [Eratosthenes of Cyrene.](https://en.wikipedia.org/wiki/Eratosthenes) There are several variants on this algorithm, and you are to implement the following one. Note that each bit represents a number, such that the first bit is 0, the second bit is 1, etc.

1. Request an integer from the user. If it is less than 3 or greater than 160,000, show an error message and go back to 1.
2. Allocate n/8 bytes of memory and fill it with all one bits. That is, put hex FF in each byte, meaning that initially we assume all numbers are prime. If n is not an even multiple of 8, round up.

And with 1 and branch if greater than 0

1. Starting with bit 2, representing the number 2, set each bit to zero that is a multiple of 2, but do not set bit 2 to zero, since 2 is prime. See example below. Next, find the next non-zero bit in the array, compute its position as a number (this will be 3) and set every third bit to zero. The array is zero origin, so as shown, the first bit is zero, then 1, etc.
2. The algorithm terminates when you have used bits up to n/2.
3. Go through the array and compute the bit position of every 1 bit, and print the position. Those will be your primes.
4. Stop

Take a number and figure out what byte offset and bit position is

Example, where the user entered 16:

Bit position: 0 4 8 12

Initial array: 1111 1111 1111 1111

First pass: 1111 0101 0101 0101 This uses 2

Second pass: 1111 0101 0001 0100 This uses 3 (what will the third pass use?)

Prints starting with bit 2: 2, 3, 5, 7, 11, 13

|  |  |
| --- | --- |
| Grading Criteria | |
| Comments and variable names | 15 |
| Correctly allocates memory and fills it | 20 |
| Correct algorithm implementation | 50 |
| Correct output | 25 |
| Total | 100 |

Additional grading criteria:

Using any algorithm other than the one provided, including writing the sieve but filling the array with integers rather than bits: -50

Use of division or multiplication. You can do this with logical operations, addition, and subtraction: -10

Printing 0, 1, or any number greater than n: -10

sbrk

**Use sysalloc call 9 to allocate memory**

**Shift right 3 to divide by 8**

**If low bits are zero then add 1**

**Use logical and to mask and check if end is zero 0+1->0**

**Use logical or/logical and to check if on or off**

**1010 =10 0001 0001=17**

**And 1 0111 AND 1 0000 0111**

**0010 =2 0000 0001=1**

**0 31**

**1111 1111 1111 1111 1111 1111 1111 1111**

**0000 1010 1010 1010 1010 1010 1010 1010 for 2**

**0000 0010 0100 1001 0010 0100 1001 0010 for 3**

**0010 0000**

**1111 1000=**

**1111 0100**

**-----To turn off**

**Start with 0010 =4 2^2 li 4 then sll2**

**0001=8 2^3 li8 then sll3**

**----To get list of primes**

**Start with 0010 bit number2**

**AND 0010**

**If !=0 then put bit number on list**

**Then sll1 and add 1 to bit number**