

## Assignment #2

**Topic(s):** C, Makefiles, Writing modular code  
**Date assigned:** Friday, September 10, 2010  
**Date due:** Friday, September 17, 2010  
**Points:** 15

The Greek astronomer Erathosthenes developed an algorithm for finding prime numbers up to some limit N in the third century B.C. The algorithm goes like this:

- 1) Write down a list of integers from 2 to N
- 2) Take the first number on the list that is not circled or crossed out and circle it because this number is prime
- 3) Cross out all remaining numbers that are a multiple of the number circled
- 4) Go to step 2 until done

Note: All circled numbers are prime

Example: Suppose the user typed in 10

**2** 3 X 5 X 7 X 9 X (circle 2 which I will show as bold and mark all multiples of 2 with an X)

2 **3** X 5 X 7 X X X (circle 3 and mark all multiples of 3 with an X)

2 3 X **5** X 7 X X X (circle 5 and mark all multiples of 5 with an X)

2 3 X 5 X **7** X X X (circle 7 and mark all multiples of 7 with an X)

2 3 X 5 X 7 X X X (the algorithm is done since no numbers remain that are not crossed out or circled)

Write a C program that implements the Sieve of Erathosthenes using an array. The user is to enter a number, N, greater than or equal to 2 and your program is to store the values from 2 to N in an array. Finally print out all of the prime numbers between 2 and N inclusive with five values per line properly aligned in columns. That is, each number is to be right-aligned in each column and each column is to take 5 places. Properly label your output. Make the array of size 1024.

In order to successfully complete this assignment, you need to write a complete modular program in Eclipse using the make facility. Here is the boiler-plate that you must use for the assignment.

Step#1: Create a project (empty C project with no auto make) called 02punetid so for me that would be 02ryand

Step#2: Inside the project create the folders **Binaries, Headers, Sources**. Remember, C is case-sensitive!!!

Step#3: Inside the Headers folder create a file called **sieve.h** with the following code.

```
#ifndef SIEVE_H_
#define SIEVE_H_

#define MAX_PRIMES 1024

void sieveLoad (int [], int);
void sieveCalculate (int [], int);
void sievePrint (int [], int);

#endif /* SIEVE_H_ */
```

Step#4: Inside the Sources folder create a file called sieve.c with the following code.

```
#include <stdio.h>
#include "../Headers/sieve.h"

/* Your function logic from Step #3 will go here */
```

Step#5: Inside the Sources folder create a file called sievedriver.c with the following code.

```
#include <stdio.h>
#include "../Headers/sieve.h"

int main (void)
{

    int sieve [MAX_PRIMES];

    /* Your program logic will go here */

    return 0;
}
```

Step #6: Create a Makefile as follows:

```
CC=gcc
CFLAGS=-g -Wall

all: sievedriver

sievedriver: Binaries/sievedriver.o Binaries/sieve.o
    ${CC} ${CFLAGS} -o sievedriver Binaries/sievedriver.o
    Binaries/sieve.o

Binaries/sievedriver.o: Sources/sievedriver.c
    ${CC} ${CFLAGS} -o Binaries/sievedriver.o -c
    Sources/sievedriver.c

Binaries/sieve.o: Headers/sieve.h Sources/sieve.c
    ${CC} ${CFLAGS} -o Binaries/sieve.o -c Sources/sieve.c

clean:
    rm sievedriver Binaries/*.o

valgrind:
    valgrind -v --leak-check=yes ./sievedriver
```

Step #7: Before writing any logic, build your project and make sure your project builds without errors.

Step #8: Commit your project using Subversion.

Step #9: Write your program one function at a time testing each function for correctness. Commit to the repository often.

In class, I will talk about how to submit this program for grading. It is somewhat tricky. You can add functions above as needed.