Assignment #2

Topic(s): C, Makefiles, Writing modular code
Date assigned: Wednesday, September 7, 2011
Date due: Wednesday, September 14, 2011
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 less than or equal to 1025 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 cs300_2_PUNetID so for me that would be **cs300_2_will4614**

Step#2: Inside the project create the folders **bin**, **src**, and **include**. Remember, C is case-sensitive.

Step#3: Inside the **include** 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 **src** folder create a file called sieve.c with the following code.

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

/* Your function logic from sieve.h will go here */
```

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

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

int main (void)
{

  int sieve [MAX_PRIMES];

  /* Your program logic will go here */

  return 0;
}
```

Step #6: Create a Makefile as follows, filling in the relevant bits marked “YOU DECIDE!”:

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

.PHONY: all clean tarball

all: sievedriver

sievedriver: bin/sievedriver.o bin/sieve.o
     YOU DECIDE!

bin/sievedriver.o: YOU DECIDE!
     ${CC} ${CFLAGS} -o bin/sievedriver.o -c src/sievedriver.c

bin/sieve.o: YOU DECIDE!
     YOU DECIDE!

clean:
     rm sievedriver bin/*.o

tarball: clean
     tar czf ../cs300_2_PUNetID.tar.gz ../cs300_2_PUNetID
```

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

Step #8: Write your program one function at a time testing each function for correctness.

You can add functions above as needed.

Don't forget to add **comments** and **test** your code thoroughly!

Be sure that sieve.c does not contain code that will crash if given bad data!

To submit your code, use the submit function on Zeus to submit **cs300_2_PUNetID.tar.gz** and turn in a color hard copy by 9:15am on the day the assignment is due.

Remember, you can only submit your project once