CSCI 340 **Computer Assignment 3** **Fall 2017**

(10 points)

Sieve of Eratosthenes

For this computer assignment, you are to write and implement an interactive C++ program to find and print all prime numbers, which are less than or equal to a given value of n, using the algorithm known as the *Sieve of Eratosthenes.*

A prime number p is an integer greater than 1 that is divisible only by 1 and p (itself). The algorithm begins by initializing a set to contain all the integers in the range 2 to n. A loop makes multiple passes over the elements in the set, using successive integer key values 2, 3, 4, … Each pass "shakes free" nonprime numbers and lets them "filter through the sieve." At the end, only the prime numbers remain.

Begin with the integer m = 2, which is the smallest prime number. The pass scans the set and removes all multiples of 2, having the form 2 * k for k >= 2. The multiples cannot be prime numbers, because they are divisible by 2. At the end of the pass, we have removed all the even numbers except 2. Next, look at the integer m =3, which is a prime number. As with value 2, remove all multiples of 3, having the form 3 * k for k >= 3. The multiples 12, 18, and so forth, are even numbers, so they have already been removed from the set. The next key integer is m = 4, which is no longer in the set, because it was removed as a multiple of 2. The pass takes no action. The process continues until it removes all multiples of prime numbers. In other words, for integer m, remove all multiples of m, having the form m * k for k >= m. The numbers that remain in the sieve are the prime numbers in the range 2 to n.

The algorithm uses an optimization feature by looking at the key values for m <= sqrt (n). However, in your implementation, test all numbers m such that m * m <= n, rather than computing the square root of n.

Programming Notes:

1. Use a set container to store the prime numbers. In the STL, a set is implemented as an *associative container*, it uses the model of a mathematical set, it stores keys that are objects of a specified data type, where duplicate keys are not allowed. To use a set in your program, you need to insert the statement: #include <set> in your header file.

2. In addition to the main ( ) routine, implement (at least) the following two subroutines in your program:

   - void sieve (set < int >& s, int n) : This routine can be used to apply the *Sieve of Eratosthenes* algorithm to remove all nonprime numbers from the integer set s = { 2, 3, …, n }.

   - void print_primes (const set < int >& s) : This routine can be used to print the elements in the integer set s (NO_ITEMS = 16 primes per line and ITEM_W = 4 spaces allocated for each prime).

3. The main ( ) routine creates an empty set of integers and prompts the user for an upper limit and calls the subroutine sieve ( ) to execute the *Sieve of Eratosthenes*. It calls the subroutine print_primes ( ) to print the prime numbers on stdout.

4. Include your header file prog3.h, by inserting the statement: #include “prog3.h” at the top of your source file prog3.cc.

5. To compile and link your program with the system library routines, execute: Make N=3. To test your program, execute: Make execute N=3. This command executes your program by getting the upper limit for prime numbers from the input file prog3.d in directory: ~cs340/progs/17f/p3, and displays the output as well as any error messages both on the terminal screen and in prog3.out. After you are done with the program, you don’t need its object and executable files any more. To delete them, execute: Make clean.

6. You can find the correct output of this program in file prog3.out in directory: ~cs340/progs/17f/p3.

7. When your program is ready, submit its source and header files to your TA by executing: mail_prog.340 prog3.cc prog3.h.