Source File: `~/2336/30/lab30.(C|CPP|cpp|c++|cc|cxx|cp)`
Input: Under control of `main` function
Output: Under control of `main` function
Value: 3

A ***prime number*** is an integer greater than 1 whose only positive divisors are 1 and the integer itself. The Greek mathematician Eratosthenes developed an algorithm, known as the ***Sieve of Eratosthenes***, for finding all prime numbers less than or equal to a given number n—that is, all primes in the range 2 through n. Consider the list of numbers from 2 through n. Two is the first prime number, but the multiples of 2 $(4, 6, 8, \ldots)$ are not, and so they are crossed out in the list. The first number after 2 that was not crossed out is 3, the next prime. We then cross out all higher multiples of 3 $(6, 9, 12, \ldots)$ from the list. The next number not crossed out is 5, the next prime, and so we cross out all higher multiples of 5 $(10, 15, 20, \ldots)$. We repeat this procedure until we reach the first number in the list that has not been crossed out and whose square is greater than n. All the numbers that remain in the list are the primes from 2 through n. Write a program that uses this sieve method to find all the prime numbers from 2 through n.

Use an `IntegerSet` (think of Lab 04) to represent the numbers 0 through n. The interface for the `IntegerSet` class has been updated and is shown in Figure 1. Some differences of particular note: the `bitVector` is kept as a `vector`, the insert member function has been renamed `set`, and the delete member function has been renamed `reset`.

```
1  #ifndef INTEGER_SET
2  #define INTEGER_SET
3
4  #include <iostream>
5  #include <vector>
6
7  using namespace std;
8
9  class IntegerSet
10 {
11   // overloaded output operator for printing IntegerSet set to
12   // output stream out
13   friend ostream& operator<<(ostream& out, const IntegerSet& set);
14  public:
15   IntegerSet(uint n = 40);                // default constructor; set consists
16                                           //   0..(n-1); set N to n; make set
17                                           //   empty
18   bool isMember(uint e) const;            // returns true if e is a member of
19                                           //   the set and false otherwise
20   IntegerSet& set(uint e);                // if e is valid and not a member of
21                                           //   the set, insert e into set
22   IntegerSet& reset(uint e);              // if e is valid and a member of
23                                           //   the set, delete e from set
24   uint size() const;                      // returns N, the # of possible
25                                           //   integers in set
26  private:
27   uint N;                                 // # of integers in set: 0..(N-1)
28   vector<uint> bitVector;                 // bit vector
29   bool isValid(uint e) const;             // 0 <= e < N
```

Figure 1. `/usr/local/2336/include/IntegerSet.h` (Part 1 of 2)

```
30    uint word(uint n) const;                // Determine index within
31                                            //   bitVector where n is located
32    uint bit(uint n) const;                 // Determine position within
33                                            //   bitVector[word(n)]
34                                            //   for element n
35  };
36
37  #endif
```

Figure 1. `/usr/local/2336/include/IntegerSet.h` (Part 2 of 2)

Your task is to write two functions, one that implements the Sieve of Eratosthenes and another that implements the overloaded output operator for the `IntegerSet` class. A sample `main` function for testing your implementation is shown in Figure 2, and a sample execution sequence is shown in Figure 3. To use the `Makefile` as distributed in class, add a target of `lab30` to `targets2srcfileswithlibrary`. The overloaded output operator function should write all of the prime numbers that remain from the sieve process. The output should be written such that one prime number is written per line.

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <vector>
4  #include <chrono>
5  #include <IntegerSet.h>
6
7  using namespace std;
8
9  void sieveOfEratosthenes(IntegerSet& prime);
10
11 int main()
12 {
13   int n;
14
15   cin >> n;
16
17   // Create an empty Integerset that can represent the range 0,1,...,n
18   IntegerSet prime(n+1);
19
20   for (uint i = 2; i < prime.size(); i++)
21     prime.set(i);
22
23   auto start = chrono::system_clock::now();
24   sieveOfEratosthenes(prime);
25   cout << prime;
26   auto stop = chrono::system_clock::now();
27   cerr << chrono::duration_cast<chrono::milliseconds>(stop-start).count()
28        << "ms" << endl;
29
30   return EXIT_SUCCESS;
31 }
```

Figure 2. `/usr/local/2336/src/lab30main.C`

```
newuser@csunix ~> cd 2336
newuser@csunix ~/2336> ./getlab.ksh 30
  * Checking to see if a folder exists for Lab 30. . .No
  * Creating a folder for Lab 30
  * Checking to see if Lab 30 has sample input and output files. . .Yes
  * Copying input and output files for Lab 30
    from folder /usr/local/2336/data/30 to folder ./30
  * Checking to see if /usr/local/2336/src/lab30main.C exists. . .Yes
  * Copying file /usr/local/2336/src/lab30main.C to folder ./30
  * Checking to see if /usr/local/2336/include/lab30.h exists. . .No
  * Copying file /usr/local/2336/src/Makefile to folder ./30
  * Adding a target of lab30 to targets2srcfileswithlibrary
  * Touching file ./30/lab30.cpp
  * Edit file ./30/lab30.cpp in Notepad++
newuser@csunix ~/2336> cd 30
newuser@csunix ~/2336/30> ls
1000.out   100000.out   10000000.out   33.out  Makefile   lab30main.C
10000.out  1000000.out  100000000.out  97.out  lab30.cpp
newuser@csunix ~/2336/30> make lab30
g++ -g -Wall -std=c++11 -c lab30main.C -I/usr/local/2336/include -I.
g++ -g -Wall -std=c++11 -c lab30.cpp -I/usr/local/2336/include -I.
g++ -o lab30 lab30main.o lab30.o -L/usr/local/2336/lib \
-Wl,-whole-archive -llab30 -Wl,-no-whole-archive -lm -lbits
newuser@csunix ~/2336/30> echo 33 | ./lab30
2
3
5
7
11
13
17
19
23
29
31
0ms
newuser@csunix ~/2336/30> echo 33 | ./lab30 > my.out
0ms
newuser@csunix ~/2336/30> diff 33.out my.out
newuser@csunix ~/2336/30> echo 97 | ./lab30 > my.out
0ms
newuser@csunix ~/2336/30> diff 97.out my.out
newuser@csunix ~/2336/30> echo 1000 | ./lab30 > my.out
0ms
newuser@csunix ~/2336/30> diff 1000.out my.out
newuser@csunix ~/2336/30> echo 10000 | ./lab30 > my.out
1ms
newuser@csunix ~/2336/30> diff 10000.out my.out
```

Figure 3. Commands to Compile, Link, & Run Lab 30 (Part 1 of 2)

```
newuser@csunix ~/2336/30> echo 100000 | ./lab30 > my.out
10ms
newuser@csunix ~/2336/30> diff 100000.out my.out
newuser@csunix ~/2336/30> echo 1000000 | ./lab30 > my.out
106ms
newuser@csunix ~/2336/30> diff 1000000.out my.out
newuser@csunix ~/2336/30> echo 10000000 | ./lab30 > my.out
1110ms
newuser@csunix ~/2336/30> diff 10000000.out my.out
newuser@csunix ~/2336/30> echo 100000000 | ./lab30 > my.out
11357ms
newuser@csunix ~/2336/30> diff 100000000.out my.out
newuser@csunix ~/2336/30>
```

Figure 3. Commands to Compile, Link, & Run Lab 30 (Part 2 of 2)