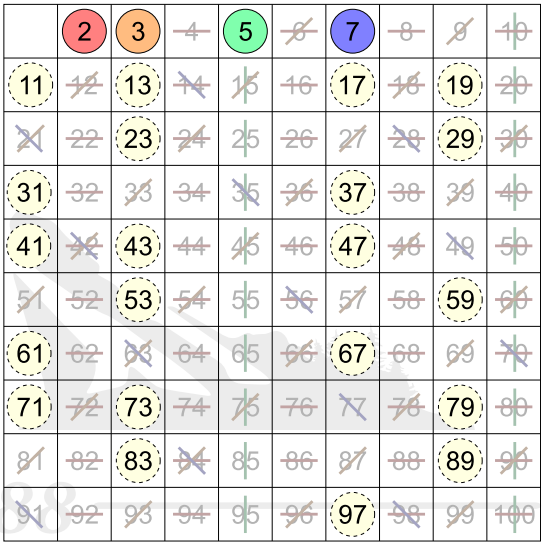
HW 7: The Sieve of Eratosthenes

The Sieve is an ancient method for finding all prime numbers up to a certain number. It is done as follows:

1. Create a list of consecutive integers from 2 to n: $(2, 3, 4, ..., n)$.
2. Initially, let p equal 2, the first prime number.
3. Strike from the list all multiples of p less than or equal to n. ($2p$, $3p$, $4p$, etc.) (To optimize, you may start striking at p^2.)
4. Find the first number remaining on the list after p (this number is the next prime); replace p with this number.
5. Repeat steps 3 and 4 until p^2 is greater than n.
6. All the remaining numbers in the list are prime.



The Sieve of Eratosthenes. Note that multiples of primes 2, 3, 5, and 7 are crossed out with a different mark. Multiples of 11 and up don't need to be crossed out, since all left-over numbers must already be prime.

The program will welcome the user, and invite him or her to enter a number. After that, all the prime numbers less than or equal to that number will be printed, separated by commas. (Be sure there's no comma at the very end.) Here is how this should look:

```
Welcome to the prime number calculator!
Please enter a maximum number: 100
The prime numbers from 2 to 100 are:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97
```

Use functions for this program. By dividing up the labor between multiple functions, development will go much more smoothly. If you try to implement the whole thing inside of your `main()` function, you will likely end up with a very large mess of spaghetti code that it'll be hard to think about coherently.

The easiest way to do this is with an array of booleans of length $(n + 1)$, where n is the maximum number. Initialize all the values to true, and then make them false as each index is determined to not be a prime number. Note that this implementation means that there will be booleans associated with the number 0 and the number 1 (which shouldn't even be considered to be prime). These booleans will be ignored.

Some further hints:

- This is the most difficult assignment you've had so far. Please don't wait until the day it's due to start it.
- Don't just calculate every number's primality separately (using a function like the one in Lab 4). You'll get severely penalized if you do so.
- Think carefully about the subtasks you can implement with each function. For example, you can make one function to create the array of booleans, one to filter out all the nonprimes, and one to print the final result. Of these three steps, filtering out the nonprimes is still a pretty big problem. You can make it easier if you make a function whose job is to filter out all the multiples of a given prime, and then calling this function from inside of a loop (so that you can call it with respect to 2, then 3, then 5...).
- Once you have a function thought out, try to test it in isolation. Start out by writing the initialization (getting the array set up), and the report (printing out the primes), without worrying about striking out non-primes. Then, when you run the program, it'll tell you that every number above 1 is prime. Then, you can work on a function to strike out multiples of a given prime. Run this with 2 as the argument. Now, the "primes" are 2, 3, 5, 7, 9, etc. (At every stage, it works a little better.) Only move on to the next step when you are 100% sure that the part you just wrote has no bugs. You should absolutely not try to program the whole thing at once, and only test it in the end. If you do that, you will be looking forward to a sleepless night as you try to debug an unholy mess that probably contains many more problems than you'd like.
- Now more than ever before, it is important to comment your code as you write it (instead of waiting until the very end), and to give your variables and functions intelligent names. If you don't, you'll end up staring at tons of code you won't understand—even though you just wrote it!

There is a more in-depth description of the Sieve at `http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes`. This also includes an animated image to show how the Sieve works.

This class should be called `Sieve`. Like always, all previous stylistic comments still hold.