Exercise - Sieving Prime Numbers

A *prime number* is any counting number greater than one that cannot be written as a product of two smaller counting numbers. For example: 2 is prime but 4 is not because $4 = 2 \times 2$. Any number that is not prime is called *composite*. There are many computational procedures for finding prime numbers. One of the oldest and simplest is called the *Sieve of Eratosthenes*. Here's how this method finds all prime numbers less than or equal to `n`:

a. We already know that `1` isn't a prime number, so start off with a vector of integers from `2` to `n`.
b. Set `p` equal to `2`, the smallest prime number.
c. Mark all the *multiples* of `p`, namely `2 * p, 3 * p, ...`, as `NOT PRIME`.
d. Find the smallest number that is both *greater* than `p` and *hasn't* been marked `NOT PRIME`. If there is no such number, go to the next step. If there is such a number, this is your new `p`. Use it to repeat step (c).
e. Congratulations: you're finished! All the numbers in your list that *aren't* market `NOT PRIME` are prime.

Here's a fully-worked example. Suppose we want to find all the prime numbers between `2` and `13`. Start with the full list.

```
2 3 4 5 6 7 8 9 10 11 12 13
```

Now *remove* all the multiples of two:

```
2 3 _ 5 6 7 _ 9 _ 11 _ 13
```

The smallest number that is bigger than `2` but isn't crossed out is `3`, so now we cross out all multiples of `3`:

```
2 3 _ 5 _ 7 _ _ _ 11 _ 13
```

The smallest number that is bigger than `3` but isn't crossed out is `5`. Now we'd cross out any multiples of `5`, but none are left. The smallest number that is greater than `5` but hasn't been crossed out is `7`. Again, there aren't any multiples of `7`. The same goes for `11` and same goes for `13`. Once we reach `13` there are no longer any values larger than `p` in our list that haven't been crossed out. So we're done!

Sieving primes by hand isn't much fun, so in this problem you'll write R code to automate the task:

1. Write R code to implement the Sieve of Eratosthenes.
2. Using your code from part 1, print out all the prime numbers between 1 and 100. You should obtain the following result:

```
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
```

Solution

There are many possibilities. Here's one, with no claim to efficiency or elegance!

```r
n <- 100
is_prime <- rep(TRUE, n) # Prime unless proven otherwise!
is_prime[1] <- FALSE # 1 isn't prime

for(p in 2:n) {
  if(is_prime[p] && (n %/% p > 1)) {
    multiples_of_p <- p * (2:(n %/% p))
    is_prime[multiples_of_p] <- FALSE
  }
}

boo <- (1:n)[is_prime]
baz <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 8
all.equal(boo, baz)
```

```
[1] TRUE
```