1. This week's Problem of the Week in Math is described as follows:

> *There are thirty positive integers less than 100 that share a certain property. Your friend, Blake, wrote them down in the table to the left. But Blake made a mistake! One of the numbers listed is wrong and should be replaced with another. Which number is incorrect, what should it be replaced with, and why?*

The numbers are listed below.

6	10	14	15	21
22	26	33	34	35
38	39	46	51	55
57	58	62	65	69
75	77	82	85	86
87	91	93	94	95

Use the fact that the "certain" property is that these numbers are all supposed to be the product of *unique* prime numbers to find and fix the mistake that Blake made.
Reminder: Code your solution in an `R` script and copy it over to this `.Rnw` file.
Hint: You may find the `%in%` operator and the `setdiff()` function to be helpful.

Solution:
Given that the certain property that all the numbers share is that they are all products of unique prime numbers, the first task at hand should be finding all possible prime numbers in the given range, which is all integers smaller than 100.

A prime number is any number whose factors are only itself and 1. The nested for loop begins by finding the factors for each number between 1 and 100, by using modular division. A number `x` is a factor of a number `y` if `x %% y == 0`.

All prime numbers will either only have one (for the number 1) or two factors (all other prime numbers). This property is used to filter out all prime numbers using `length(all.factors) <= 2`. I added `!i %in% all.primes` to catch any duplicates.

The property that all the numbers in Blake's list share is that they are products of unique prime numbers, specifically numbers less than 100. The nested for loop first begins by calculating all possible prime products. By using `all.primes[k] != all.primes[k]` and `prime.product < 100`. I used `setdiff()` to eliminate all duplicates and other nonunique prime products, including prime products of the form $1*$ a prime number.

After this, we can use `setdiff()` to compare the actual set of unique prime products to Blake's list. `setdiff(sample.nums, prime.products)` find the elements in `sample.nums` not found in `prime.products`. Flipping the order of the parameters does the same process in reverse, finding elements of `prime.products` not in `sample.nums`. This gives us the answers to which number in Blake's list is wrong, and which number should be included in his list, respectively.

```r
# information from problem
all.nums = 1:100
sample.nums = c(6, 10, 14, 15, 21, 22, 26, 33,
                34, 35, 38, 39, 46, 51, 55, 57,
                58, 62, 65, 69, 75, 77, 82, 85,
                86, 87, 91, 93, 94, 95)
# finding all primes between 1 and 100
all.primes = c()
for (i in 1:length(all.nums)){
  all.factors = c()
  for (j in 1:i){
    if (i %% j == 0){
     all.factors = c(all.factors, j)
    }
  }
  if (length(all.factors) <= 2 & !i %in% all.primes){
    all.primes = c(all.primes, i)
  }
}
# finding all unique prime products under 100
prime.products = c()
for (k in 1:length(all.primes)){
  for(l in 1:length(all.primes)){
    prime.product = all.primes[k] * all.primes[l]
    if (all.primes[k] != all.primes[l]
        & prime.product < 100){
      prime.products = setdiff(c(prime.products, prime.product), all.primes)
    }
  }
}
# code to find which number is out of place
setdiff(sample.nums, prime.products)
```

```
## [1] 75
```

```r
# code to find which number should be in
# of 75
setdiff(prime.products, sample.nums)
```

```
## [1] 74
```