Lab: Sequence Comprehensions

In the background section patterns for using comprehensions consisted of the following three steps, in order: 1) values are assigned to variables; 2) those variables are used in a comprehension, the value of which is saved in a variable; 3) that variable contains the solution and some operation on the variable reveals it. For example, a group of three `let` statements, including a comprehension, for reversing a sequence could look like this:

```
Cryptol> let lst = [16,42,23,77]
Cryptol> let z = [lst] # [[b]#(tail (a >> 1)) | a <- z | b <- lst ]
Cryptol> let rev = z ! 0
Cryptol> rev
[77,23,42,16]
```

Solutions to the exercises below will have a different, far more convenient and useful form where the number of `let` statements is reduced to one. For the above this looks like the following:

```
Cryptol> let rev lst = z ! 0
                where
                   z = [lst] # [[b]#(tail (a >> 1)) | a <- z | b <- lst ]
```

With the above, the reverse of a given sequence may be found like this:

```
Cryptol> rev [16,42,23,77]
[77,23,42,16]
```

Note: when entering such a `let` statement into the Cryptol console, the statement must be written as a single line or it must be broken up into multiple line, each ending in the character `\` without a following character, not even a space. Alternatively, write the expression on multiple lines in a file of name ending in `.cry`. Then the file may be loaded like this:

```
Cryptol> :l named_file.cry.
```

Exercise 1:
Given a sequence `lst` of non-negative integers, find the value of the greatest number in `lst` and the earliest position of that number, from the left, in `lst`.

Exercise 2:
Given a sequence `lst` of integers and an integer `n`, determine whether `n` is a member of `lst`.

Exercise 3:
Given a sequence `lst` of non-negative integers and an integer `n`, if `n` is a member of `lst` then remove the leftmost `n` in `lst` otherwise `lst` is not changed. **Note:** because Cryptol is so strongly typed, the possibility of returning with a sequence of length that cannot be determined until runtime is not allowed. A simple way to work around this difficulty is to append `-1` to `lst` in case `n` is removed from it.

Exercise 4:
Given two sequences `x` and `y` of non-negative integers, determine whether `x` is a permutation of `y`.

Exercise 5:
Determine whether a sequence of non-negative integers is also non-decreasing.