```
#################################################
######    VECTOR-BASED PROGRAMMING EXERCISES
#################################################

# Can find sum of the squares of a vector
# beginning at 1 and going to n by using a loop

n <- 50
S <- 0
for (i in 1:n) {
    S <- S + i^2
}
S

# Of course, R has its own vector operations
# that does the same thing, only 'better':

sum((1:n)^2)

# Ignoring that for a moment, write a function
# called sum.squares() that, when called with no
# arguments (i.e. 'sum.squares()') performs the
# the tasks in lines 8-13 above. Call your
# function to test it.

sum.squares <- function() {
  . . . <your work goes here>
}

sum.squares()

# Now modify the sum.squares() function (call the
# modified function sum.squares1) so that it now
# accepts one argument which is the high number (n)
# in the vector to be squared. Assume the vector
# still begins at 1. Test your function.

sum.squares1 <- function(x) {
  . . . <your work>
  }
  return(S)
}

sum.squares1(50)

# Now modify the sum.squares1() function again
# to accept a second argument, the power to
# raise the elements of the vector to (the
# exponent) before they are summed. Test your
# function.

sum.squares2 <- function(x,y) {
  . . . <your work>
  }
  return(S)
}

sum.squares2(50,2)

# If you did not do this already, modify your
# function so that the second argument (the
# exponent) has a default value of 2 if the
# user forgets to call that argument. Test
# your function.

sum.squares3 <- function(x,y=2) {
  . . . <your work>
  }
  return(S)
}

sum.squares3(10,5)

# What happens when you try to enter the vector
# c(1,2,3) as the first argument (the value for
# n) to your function? Re-write your function so
# that it will accept a vector input and then
# raise each element of that vector to the power
# indicated (should still have two arguments).
# Test your function.
```

```
sum.squares(c(1,2,3))

# We get an error

sum.squares4 <- function(x,y=2) {
  . . . <your work>
  }
  return(S)
}

sum.squares4(3:10,5)

# Modify it again so that it also prints
# out the exponentiated value of each element
# of the input vector, as well as the final
# summed value of all of the exponentiated
# elements of the vector. Test it.
```