```
# quiz1 solution -- there are often many different for each problem
# I've left some "scaffolding" in place to show how I set up and/or verify my solutions

# 1.  Create a vector that contains 20 numbers.
# (You may choose whatever numbers you like,
# but make sure there are some duplicates.)

numVec <- c(1:8,7:14,7:10)
numVec

# 2. Use R to convert the vector from question 1 into a character vector.
charVec <- as.character(numVec)
charVec

# 3.  Use R to convert the vector from question 1 into a vector of factors.
numFactors <- as.factor(numVec)
numFactors

# 4.  Use R to show how many levels the vector in the previous question has.
nlevels(numFactors)

# alternative solution
length(unique(numFactors))

#5 Use R to create a vector that takes the vector from
# question 1 and performs on it the formula 3x^2 - 4x + 1

numVec <- 3 * numVec ^ 2 - 4 * numVec + 1
numVec


#6. Create a named list. That is, create a list with several elements
# that are each able to be referenced by name.

namedlist <- list(rank = 1:5, players = c("Ada Lovelace", "Grace Hopper"))

namedlist$players[2]
```

```
# 7. Create a data frame with four columns - one each character,
# factor (with three levels), numeric, and date.
# Your data frame should have at least 10 observations (rows).

# source: http://www.buzzfeed.com/kateaurthur/all-85-best-picture-oscar-winners-ranked#1gjwibh
# I took a lot of liberties in classifying genres

title <- as.character(c("Unforgiven","The Deer Hunter", "It Happened One Night",
             "The Bridge on the River Kwai", "Lawrence of Arabia",
             "The Silence of the Lambs", "The Godfather Part II", "Casablanca",
             "The Godfather", "All about Eve"))
genre <- c("Western", "Drama", "Comedy",
             "Drama", "Drama", "Drama", "Drama", "Drama", "Drama", "Drama")
wins <- as.numeric(c(4, 8, 5, 7, 7, 5, 6, 3, 3, 6))
release <- as.Date(c("1992-1-1", "1978-1-1", "1934-1-1", "1957-1-1", "1962-1-1",
          "1991-1-1", "1974-1-1", "1943-1-1", "1972-1-1", "1950-1-1"))

df <- data.frame(title, genre, wins, release)
df$title <- as.character(df$title)
str(df)

# 8.  Illustrate how to add a row with a value for the factor column that isn't already in the list
of levels.
# (Note: You do not need to accomplish this with a single line of code.)

# Task:  Add "West Side Story", "Musical", 10, "1961-1-1"

dfnew <- data.frame(title="West Side Story", genre="Musical",
                  wins=10, release="1961-1-1")
str(dfnew)
dfnew$release <- as.Date(dfnew$release)
dfnew$title <- as.character(dfnew$title)

str(dfnew)
dfnew$title
str(dfnew)

df <- rbind(df, dfnew)
df
str(df)
```

```
# 9.  Show the code that would read in a CSV file called temperatures.csv
#      from the current working directory.

getwd()

df <- read.table(file = "temperatures.csv", header = TRUE, sep = ",")
tail(df)
str(df)

# alternative:
df <- read.csv(file = "temperatures.csv")
df

#10.  Use a loop to calculate the final balance, rounded to the nearest cent,
#      in an account that earns 3.24% interest compounded monthly after six years
#      if the original balance is $1,500.

principal <- 1500
annual_interest_rate <- 0.0324
compounding_periods_per_year <- 12
years <- 6

# loop below is less efficient than if we pre-calculated compounding_periods_per_year * years
#   and annual_interest_rate / compounding_periods_per_year
for (period in 1:(compounding_periods_per_year*years))
{
  principal <- principal + (principal * annual_interest_rate / compounding_periods_per_year)
}
print (principal)  # prints 1821.396
sprintf("%.2f", principal) #formats as 1821.40

# alternative: calculating interest without a loop
principal <- principal * ( 1 + (annual_interest_rate/compounding_periods_per_year)) ^
  (compounding_periods_year * years)
print (principal)
```

```
# 11.  Create a numeric vector of length 20 and
# then write code to calculate the sum of every third element of the vector you have created.
nvect <- 1:20
print (nvect)  # check by hand: should sum to 63
nvect %% 3
nvect %% 3 == 0

sum(nvect)

sum(nvect[nvect %% 3 == 0])   # answer; prints 63

# alternative solution
nvect <- c(1:20)
sum(nvect[seq(0,length(nvect),3)])

# 12 Use a for loop to calculate sum of i = 1 to 10 of x^i  for the value x=2

x <- 2
sum <- 0
for (i in 1:10)
{
  sum <- sum + x ^ i
}
print (sum)

# 13.  Use a while loop to accomplish the same task as in the previous exercise.

x <- 2
sum <- 0
i <- 1
while (i <= 10)
{
  sum <- sum + x ^ i
  i <- i + 1
}
print (sum)

# 14.  Solve the problem from the previous two exercises without using a loop.

x <- 2
print(sum(x^(1:10)))

# an alternative solution: observe that this could also be written as 2 ^ (n+1) - 2

print(2^(10+1)-2)
```