#---------------

# QUESTION 1

#---------------

# Assume that x is a variable that contains a positive integer (i.e. a whole number).

# Write a command that produces x number of copies of the value stored in x. For example

#

# EXAMPLE 1

# > x = 3

# > YOUR ANSWER GOES HERE

# [1] 3 3 3

#

# EXAMPLE 2

# > x = 5

# > YOUR ANSWER GOES HERE

# [1] 5 5 5 5 5

###########

###########

# ANSWER

###########

###########

x = 3 # set x to 3

rep(x,x) # three 3's

x=5 # set x to 5

rep(x,x) # five 5's

#---------------

# QUESTION 2

#---------------

# x contains a positive number that is 2 or more.

# Write a command that creates a vector that contains

# x number of copies of the value stored in x followed by

# x+1 number of copies of x+1.

#

# EXAMPLE 1

# > x = 2 # set x to 2

# > YOUR ANSWER GOES HERE # generate two 2's and three 3's

# [1] 2 2 3 3 3

#

# EXAMPLE 2

# > x=5 # set x to 5

# > YOUR ANSWER GOES HERE # generate five 5's six 6's ( same command )

# [1] 5 5 5 5 5 6 6 6 6 6 6

###########

###########

# ANSWER

###########

###########

x = 2 # set x to 2

c( rep(x,x), rep(x+1,x+1) ) # two 2's three 3's

x=5 # set x to 5

c( rep(x,x), rep(x+1,x+1) ) # five 5's six 6's ( same command )

#---------------

# QUESTION 3

#---------------

# x contains a number (or a vector of several numbers).

# Write a command that rounds down the value(s) of x to the nearest 10.

# NOTE - The round function does NOT do this. Instead, try to approach the

# problem a different way by using a combination of math and other functions

# that we learned about.

#

# EXAMPLE 1

# > x <- 32

# > YOUR ANSWER GOES HERE

# [1] 30

#

# EXAMPLE 2

# > x <- -5188.25 # this one is negative

# > YOUR ANSWER GOES HERE

# [1] -5190

#

# EXAMPLE 3

# > x <- 2.4

# > YOUR ANSWER GOES HERE

# [1] 0

#

# EXAMPLE 4

# > x <- c(32, 198.2, 2.4, -198.2)

# > YOUR ANSWER GOES HERE

# [1] 30 190 0 -200

#

# HINT - Remember that any number divided by 10 moves the decimal point

# to the left. For example 123/10 is 12.3

#

# Remember, multiplying an number by 10 moves the decimal point to

# the right. For example, 123.0 \* 10 is 1230

###########

###########

# ANSWER

###########

###########

#................

# ONE ANSWER

#................

x <- 32

floor(x/10) \* 10

x <- -5188.25 # this one is negative

floor(x/10) \* 10

x <- 2.4

floor(x/10) \* 10

x <- c(32, 198.2, 2.4, -198.2)

floor(x/10) \* 10

#................

# A SECOND ANSWER

#................

x <- 32

x %/% 10 \* 10

x <- -5188.25 # this one is negative

x %/% 10 \* 10

x <- 2.4

x %/% 10 \* 10

x <- c(32, 198.2, 2.4, -198.2)

x %/% 10 \* 10

#---------------

# QUESTION 4

#---------------

# x is a variable that contains a positive number, the code below shows how to

# use the seq function to produce the first x multiples of 5.

# Write a different command that does NOT use the seq function but that

# accomplishes the same result.

#

# SOLUTION THAT USES THE seq FUNCTION

#

# > x = 3 # set x to 3

# > seq(from=5, to=5\*x, by=5) # show first 3 multiples of 5 (type x instead of 3)

# [1] 5 10 15

#

# > x = 6 # set x to 6

# > seq(from=5, to=5\*x, by=5) # show first 6 multiples of 5 (same command as above)

# [1] 5 10 15 20 25 30

#

# NOW DO IT A DIFFERENT WAY ...

#

# EXAMPLE 1

# > x = 3

# > YOUR ANSWER GOES HERE

# [1] 5 10 15

#

# EXAMPLE 2

# > x = 6

# > YOUR ANSWER GOES HERE

# [1] 5 10 15 20 25 30

#

###########

###########

# ANSWER

###########

###########

x = 3

1:x \* 5

x = 6

1:x \* 5

#---------------

# QUESTION 5 (a)

#---------------

# x contains a positive number.

# Write a command that produces the values

# 1\*2, 2\*3, 3\*4, etc .. up until x\*x+1

# Do this using the seq function

#

# EXAMPLE 1

# > x = 3 # show the three values: 1\*2 2\*3 3\*4

# > YOUR ANSWER GOES HERE

# [1] 2 6 12

#

# EXAMPLE 2

# > x = 5 # show the five values: 1\*2 2\*3 3\*4 4\*5 5\*6

# > YOUR ANSWER GOES HERE

# [1] 2 6 12 20 30

###########

###########

# ANSWER

###########

###########

x = 3

seq(1, x) \* seq(2, x+1)

x = 5

seq(1, x) \* seq(2, x+1)#---------------

# QUESTION 5 (b)

#---------------

# (b) Same question as above, but do it in a different way WITHOUT using the seq function

###########

###########

# ANSWER

###########

###########

x = 3

(1:x) \* 2:(x+1)

x = 5

(1:x) \* 2:(x+1)

#---------------

# QUESTION 6

#---------------

# Assume that x is an integer that is 2 or more.

# Write a command that generates the sequence of numbers

# 1 -2 3 -4 5 -6 etc ... x (or -x)

# The numbers should alternate between positive and negative numbers.

# The final value should be the positive or negative value of x.

# NOTE: It is OK if warnings are generated as long as you get the right output.

#

# HINT: Remember, if you multiply a number by 1 you just get back the same

# number but if you multiply a number by -1 you get back the negative

# version of the number.

###########

###########

# ANSWER

###########

###########

x=2

c(1,-1) \* (1:x)

x=7

c(1,-1) \* (1:x) # this will generate a "warning" but that is OK for this assignment (see the instructions)

x=8

c(1,-1) \* (1:x)

#---------------

# QUESTION 7

#---------------

# BACKGROUND INFO

#

# The following is an example of an "infinite series".

# It can theoretically go on forever ...

#

# 6/4^1 6/4^2 6/4^3 6/4^4 + ...

#

# The "sum" of this infinite series is :

# 6/4^1 + 6/4^2 + 6/4^3 + 6/4^4 + ...

#

# which is equal to

#

# 6/4 + 6/16 + 6/64 + 6/256 + ...

#

# which is equal to

#

# 1.5 + .375 + .09375 + .0234375 + ...

#

# We cannot add up all of the values in the sum since there are an infinite

# number of values in the sum. However, we can add up the first several

# values up to a certain point. These are called "partial sums".

#

# The first "partial sum" of this infinite series is : 1.5

# The 2nd partial sum is: 1.5 + .375 = 1.875

# The 3rd partial sum is: 1.5 + .375 + .09375 = 1.96875

# The 4th partial sum is: 1.5 + .375 + .09375 + .0234375 = 1.9921875

# etc ...

#

# As more and more values are added to the partial sum, the sum gets closer and closer

# the number 2. We say that the sum "converges" to 2.

#

# QUESTION

# x is a positive integer. Write a command to calculate the

# partial sum of the first x values of the infinite series shown above.

#

# EXAMPLE 1

# > x = 2 # automatically add the first 2 values, i.e. 1.5 + .375

# > YOUR ANSWER GOES HERE

# [1] 1.875

#

# EXAMPLE 2

# > x = 3 # automatically add the first 3 values, i.e. 1.5 + .375 + .09375

# > YOUR ANSWER GOES HERE

# [1] 1.96875

#

# EXAMPLE 3

# > x = 9 # automatically add the first 9 values in the series

# > YOUR ANSWER GOES HERE

# [1] 1.999992

#

# NOTE - by default if the sum gets "really close" to 2, then R will display

# the result as 2, even though the actual sum is a bit less than 2. For example

# if x is 11 then by default R will display the sum of the first 11 positions

# in the infinite sum as 2 even though the sum is actually a bit less than 2.

# The reason for this is by default, R displays only up to 7 digits after

# a decimal point. However, this can be changed with R's "options" command

# (type ?options to get help on this command).

#

# For example 1/7 is a non-terminating decimal, ie.

# 1/7 is mathematically equal to 1.142857142857142857142857142857...etc

# The 142857 after the decimal point just keeps repeating infinitely.

# However, by default if you type 1/7 in R you will see 0.1428571 since

# R will by default display only 7 digits after the decimal point. However, if

# you type options(digits=20) and then

# type 1/7 you should see the following: 0.14285714285714285

# You can revert back to the original number of digits by typing

# options(digits=7).

#

# Similarly for our problem if you type options(digits=20) and then run

# your command for x=11, you should see 1.9999995231628418

###########

###########

# ANSWER

###########

###########

options(digits=7)

x=2

sum(6/4^(1:x))

x=3

sum(6/4^(1:x))

x=9

sum(6/4^(1:x))

#---------------

# QUESTION 8 (a)

#---------------

# x is an integer that contains 2 or more.

# Generate x number of values in the sequence

#

# 2 4 6 8 ...

#

# EXAMPLE 1

# > x=4

# > YOUR ANSWER GOES HERE

# [1] 2 4 6 8

#

# EXAMPLE 2

# > x=6

# > YOUR ANSWER GOES HERE

# [1] 2 4 6 8 10 12

#

# HINT - use the seq function. Make sure that you use the different

# arguments to the seq function in the best possible way

# to solve this problem.

###########

###########

# ANSWER

###########

###########

x=4

seq(2,length.out=x,by=2)

x=6

seq(2,length.out=x,by=2)

#---------------

# QUESTION 8 (b)

#---------------

# x is an integer that contains 2 or more.

# Generate a vector that contains the first x number of values in the series

#

# 2\*3\*4 4\*5\*6 6\*7\*8 8\*9\*10 ... etc

# i.e. R should show

# 24 120 336 720 ... etc

#

# EXAMPLE 1

# > x=4

# > YOUR ANSWER GOES HERE

# [1] 24 120 336 720

#

# EXAMPLE 2

# > x=6

# > YOUR ANSWER GOES HERE

# [1] 24 120 336 720 1320 2184

#

# HINT: refer to your answer to part (a).

# Think about how you can use the recycling rule to build on that answer.

###########

###########

# ANSWER

###########

###########

x=4

seq(2,length.out=x,by=2) \* seq(3,length.out=x,by=2) \* seq(4, length.out=x,by=2)

x=6

seq(2,length.out=x,by=2) \* seq(3,length.out=x,by=2) \* seq(4, length.out=x,by=2)

#---------------

# QUESTION 8 (c)

#---------------

# As you may know, PI, has a never-ending non-repeating number of

# digits after the decimal point. The first few digits of pi are 3.14159... etc.

# This webpage contains the value of pi to one million decimal places - https://www.piday.org/million/

# It is impossible to calculate the "last" digit of pi since you can always

# keep calculating more decimal places.

# There are many different ways to mathematically calculate pi to greater and

# greater precision. This webpage shows a few methods: https://www.wikihow.com/Calculate-Pi

#

# For this question you will approximate the value of PI by using

# partial sums of an infinite series known as the Nilakantha Series

# If you're curious see these links:

# - http://www.maeckes.nl/Formule%20voor%20pi%20(Nilakantha)%20GB.html

# - https://www.youtube.com/watch?v=3U7Hfz37GqA

# This Nilakantha series consists of the values shown below

# Notice the "3" in the first position.

# Notice the pattern of values in the denominators of the 2nd and subsequent terms.

# Notice that the terms alternate between positive and negative.

#

# 3 + 4/(2\*3\*4) - 4/(4\*5\*6) + 4/(6\*7\*8) - 4/(8\*9\*10) + 4/(10\*11\*12) - ... etc

#

# Set x to the number of terms in the Nilakantha Series to add to the initial "3".

# Note, a simple way to do this problem will cause problems when x=0 or x=1.

# Therefore, you can ignore those cases.

# (We can discuss later in class why x=0 and x=1 may cause problems).

#

# EXAMPLE

# > x = 2 # Two terms added to 3, i.e. 3 + 4/2\*3\*4 - 4/4\*5\*6

# > YOUR ANSWER GOES HERE

# [1] 3.1333333

#

# EXAMPLE

# > x = 3 # Three terms added to 3, i.e. 3 + 4/2\*3\*4 - 4/4\*5\*6 + 4/6\*7\*8

# > YOUR ANSWER GOES HERE

# [1] 3.145238

#

# EXAMPLE

# > x = 4 # Four terms added to 3, i.e. 3 + 4/2\*3\*4 - 4/4\*5\*6 + 4/6\*7\*8 - 4/8\*9\*10

# > YOUR ANSWER GOES HERE

# [1] 3.139683

#

# EXAMPLE

# > x = 50 # Fifty terms added to 3

# > YOUR ANSWER GOES HERE

# [1] 3.141591

#

# HINT 1: Do NOT use R's "pi" variable to do this question.

#

# HINT 2: In part (b) above, you generated the sequence 2\*3\*4, 4\*5\*6, 6\*7\*8, etc

# Notice that that sequence is in the denominators of the fractions in this

# question.

#

# HINT 3: You already figured out how to do the alternating + and - signs

# in one of the questions above.

#

# HINT 4: Try different things and build up your answer slowly.

#

# HINT 5: Try to figure out how to automatically generate the terms and display them.

# Then if the terms you are generating are correct,

# you can simply change your answer to use the sum function.

#

# To see what our code should eventually produce, we can "cheat" and

# simply type out the first few terms (without automatically generating

# them as you need to do) :

#

# > c(3 , +4/(2\*3\*4) , -4/(4\*5\*6) , +4/(6\*7\*8) , -4/(8\*9\*10))

# [1] 3.000000000 0.166666667 -0.033333333 0.011904762 -0.005555556

#

# summing them gives us

# > sum(c(3, +4/(2\*3\*4), -4/(4\*5\*6), +4/(6\*7\*8), -4/(8\*9\*10)))

# [1] 3.139683

#

# Now, try to write an R command to automatically produce the vector shown above

# when x is set to 4.

#

# HINT 6: Make sure that your answer works for x=2, x=3,etc (don't worry about x=1 or x=0)

###########

###########

# ANSWER

###########

###########

options(digits=7)

x = 2

3 + sum(c(1,-1) \* (4 / (seq(2,by=2,length.out=x) \* seq(3,by=2,length.out=x) \* seq(4,by=2,length.out=x) )))

x = 3

3 + sum(c(1,-1) \* (4 / (seq(2,by=2,length.out=x) \* seq(3,by=2,length.out=x) \* seq(4,by=2,length.out=x) )))

x = 4

3 + sum(c(1,-1) \* (4 / (seq(2,by=2,length.out=x) \* seq(3,by=2,length.out=x) \* seq(4,by=2,length.out=x) )))

x = 50

3 + sum(c(1,-1) \* (4 / (seq(2,by=2,length.out=x) \* seq(3,by=2,length.out=x) \* seq(4,by=2,length.out=x) )))

############

############

# END OF HW

############

############