

Total Points: 26/30 + 1 = 27 points

```
##
## Orysyia Stus
## Assignment 2
##

##
## Problem 1
##
## The purpose of this problem is detecting extreme values. Write a
function, extreme, to detect extreme values. You can consider values
greater than or less than 3 times standard deviation from the mean as
extreme values.
# This function will take only one argument, which is a numeric vector.
# If extreme values were found, say 5, print \There are 5 extreme values
found." If extreme values were not found, print \There are no extreme
values."
# Generate 1000 numbers following standard normal distribution and use
this function to test the extreme function.
extreme = function(x) {
  for(i in x) {
    count = 0
    m      = mean(x)
    sd     = 3*sd(x)
    if(11<(m-sd) || 11>(m+sd)) {
      count = count + 1
    }
  }
  if(count>0) {
    print(paste("There are ", count, " extreme values found.", sep = ""))
  } else print("There are no extreme values.")
}


test = rnorm(1000)
extreme(test)

##
## Problem 2
##
## Write a function, calCS, to perform the following task:
# Calculate the area of a circle (AC =  $r^2$ ), the circumference of a circle
(CC =  $2r$ ), the volume of a sphere (V S =  $\frac{4}{3}r^3$ ), or the area of a sphere
(AS =  $4r^2$ )
# The rst argument is either AC, CC, VS, or AS to determine which
calculation needs to be performed
# The value that the rst argument takes can contain either lower, upper,
or mixed cases of letters (Use the toupper function)
# If the values of the argument are not AC, CC, VS, or AS, stop the
function and write your method is not supported'
# The second argument is the radius (r). Make sure to use if ... else
statements for this problem
calCS = function(method, r) {
```


No need to use loop here.

See my solution

```
method = toupper(method)
if (method == "AC") return (pi*r*r)
else if (method == "CC") return (2*pi*r)
else if (method == "VS") return ((4/3)*pi*r*r*r)
else if (method == "AS") return (4*pi*r*r)
else stop ("your method is not supported")
}
# quick test
calCS("ac", r)
```




```
##
## Problem 3
##
## For circles with radii 5, 10, 15, 20, and 25, write a loop to
calculate the the area of the circle by using the function that you just
wrote. In previous function, to calculate the area of a circle with
radius = 5,
## we can simply write calCS(`AC`, 5); however, when you insert the
function inside the loop, you need to explicitly use the print function.
For example, print (calCS(`AC`, 5))
x = c(5,10,15,20,25)
for (i in x) {
  print (calCS("AC", i))
}
```



```
##
## Problem 4
##
## Based on the painters data set from the MASS library:
# Create a data set which contains observations with Colour >=17 and
School equals \D".
library(MASS)

d1 <- subset(painters, Colour >= 17 & School == "D")
d1
# Create a data set that contains only Da Udine and Barocci.
d2 <- painters[c(1, 17), ]
d2
# Create a data set which contains observations with Colour >=17 and
School equals \D", but only with the Composition and Drawing variables.
d3 <- d1[, c("Composition", "Drawing")]
d3
# Create a categorical variable Comp.cat with three approximate equal
levels based on Composition.
d4 <- split(painters, cut(painters$Composition, 3))
d4
```



```
##
## Problem 5
##
## For this problem, we will use the following data, which is in the wide
form:
# After creating this data set, transform it into the long form.
# Then transform the long form back to the wide form.
```

```

Program <- c('CONT', 'RI', 'WI')
s1 <- c(85, 79, 84)
s2 <- c(85, 79, 85)
s3 <- c(86, 79, 84)
s4 <- c(85, 80, 83)

wide.data <- data.frame(Program, s1, s2, s3, s4)
wide.data

long = reshape(wide.data, varying= list(c("s1", "s2", "s3", "s4")),
v.names = "Score", timevar="TIME", idvar="ID", direction="long")
long

widel = reshape(long, varying= list(c("s1", "s2", "s3", "s4")),
v.names="Score", timevar="TIME", idvar="ID", direction="wide")
widel

##
## Problem 6
##
## Write a function, called stackDataInList. This function only takes one
argument, alist, which is a list of data.frame. The number of elements in
the list varies. In this function, you will stack the data frames
# on top of each other and return one single data frame. There are many
solutions to solve this problem; however, please make sure to use a loop
within the function to solve this problem. Without using a loop
# will not receive credits for this problem.
stackDataInList = function(alist) {
  for (df in alist) {
    as.list(df)
  }
  return (rbind(df))
}

#For testing
load("C:/Users/Orysy/Desktop/Introduction_to_R_Programming/Assignment2_W
5-7/datList.RData")
datList
stackDataInList(datList[1])
stackDataInList(datList[c(1,3,4)])
stackDataInList(datList)
stackDataInList(datList)

```

It doesn't work for a list with length greater than one.

Points: -4