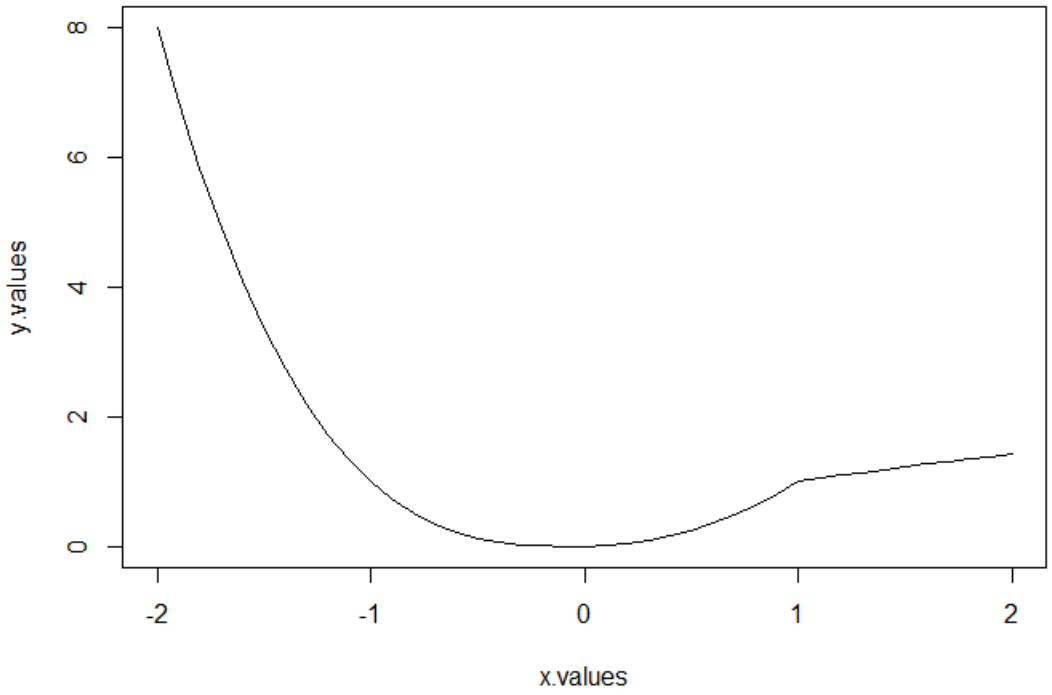
```
#Assignment# 02

#Statistical Programming with R

#Date: 02/26/2018

#Author: Casey Carr

# Questions: 1, 5, 7, 12, 13

# Question 1: Consider the function y = f(x) such that y = -x^3 for x<= 0, y = x^2 for x in (0,1], y = sqrt(x) for x > 1

# Supposing that you are given x, write an expression for y using if-statements.

# input:

x.values <- seq(-2,2,by = 0.1)

# for each x calculate y

n <- length(x.values)

y.values <- rep(0,n)

for (i in 1:n) {

  x <- x.values[i]

  if (x<=0) {

  y = -x^3

} else if (x > 0 && x <= 1) {

  y = x^2

} else {

  y = sqrt(x) }

 y.values[i] <- y }

# output

plot(x.values, y.values, type = 'l')

# Does the plot look like figure 3.2? Do you think f has a derivative at x = 1? x = 0?

# The derivative of f at x = 1 does NOT exist since there exists a "corner" at that point

# The derivative at x = 0 EXISTS
```



Question 5: To rotate a vector (x,y)^T widdershins (anticlockwise) by theta radians, you premultiply it by the matrix (cos(theta) -sin(theta); sin(theta) cos(theta)), where ';' indicates a carriage return to the next row.

Write an R program that does this for you

```
rotateVector <- function(vectorToRotate, thetaRadians) {
 rotationMatrix <- matrix(c(cos(thetaRadians), -sin(thetaRadians), sin(thetaRadians), cos(thetaRadians)),
 byrow = TRUE, ncol = 2)
 result <- rotationMatrix %*% vectorToRotate
 return(result)
}
```

enter any vector in R2 and theta in radians to return the rotated solution

```
rotateVector(c(1,2),pi/2)
```

```
     [,1]
[1,]   -2
[2,]    1
```

Question 7: How would you find the sum of every third element of a vector X?

vectorA = c(1,2,4,5,6,7,9,10,11,12)

sum(vectorA[(1:length(vectorA))%%3==0]) # iterates through vectorA and only sums the elements at indices divisible by 3

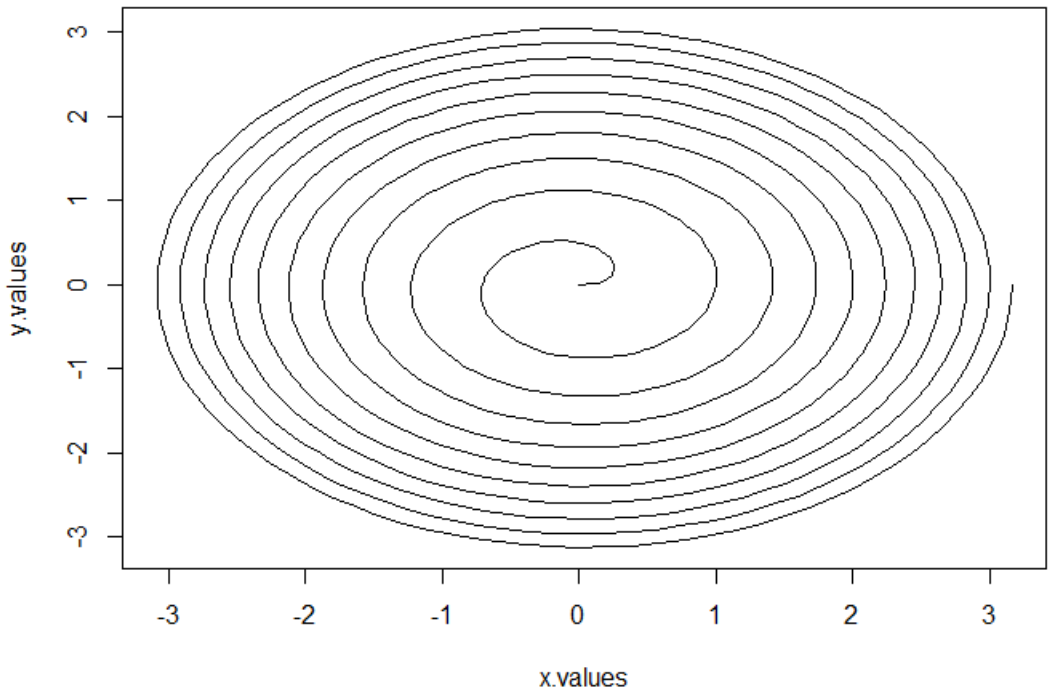
[1] 22

' Question 12: The dice game craps is played as follows. The player throws two dice, and if the sum is seven or eleven, then he wins. If the sum is two, three, or twelve, then he loses. If the sum is anything else, then he continues throwing until he either throws that number again (in which case he wins) or he throws a seven (in which case he loses).

Write a program to simulate a game of craps. You can use the following snippet of code to simulate the roll of two (fair) die: x <- sum(ceiling(6*runif(2))) ‘

```
playDice <- function() {
  currentPoint <- sum(ceiling(6*runif(2)))
  if (currentPoint == 7 || currentPoint == 11) {
    return("First roll win!")
  } else if (currentPoint == 2 || currentPoint == 3 || currentPoint == 12) {
    return("First roll crap out! You lose.")
  } else {
    point <- 0
    while (point != currentPoint) {
      point <- sum(ceiling(6*runif(2)))
      if (point == 7) {
        return("You lose!")
      } else if (point == currentPoint) {
        return("You win!")
      }
    }
  }
}
playDice()  [1] "You lose!"
```

```
' Question 13: Suppose that (x(t), y(t)) has polar coordinates (sqrt(t),2pi*t). Plot (x(t),y(t)) for t in [0,10]
        Note: in Polar Coordinates we use (r,theta), thus r = sqrt(t), theta = 2*pi*t
        Therefore, x = rcos(theta) = sqrt(t)*cos(2*pi*t) and
            y = rsin(theta) = sqrt(t)*sin(2*pi*t)  '
t.values <- seq(0,10, by = 0.01)
n <- length(t.values)
x.values <- rep(0, n) #initalize each of these vectors
y.values <- rep(0, n)
for (i in 1:n) {
  x.values[i] <- sqrt(t.values[i])*cos(2*pi*t.values[i])
  y.values[i] <- sqrt(t.values[i])*sin(2*pi*t.values[i])
}
plot(x.values,y.values, type = "l")
```



PAST SUBMISSION REVIEW:

Anusha Sunkara (bottom)

1. For #1, there is no need to return x.values , y.values and no need to assign to prob1 – I would advise to try to make code as clean as possible
2. For #5, a more scalable approach would be to automate the vector rotation via a function call
3. For #7, nice one-liner for the sum but could have been achieved without for-loop via vector slicing.
4. Fort #12, code is difficult to follow. Be sure to use proper indentation to enhance readability. I would advise to use descriptive variable names.

Brett Statser (bottom)

1. For #7, generating a matrix just to extract the 3rd column is a cumbersome approach. A more concise approach would be preferred such as vector slicing.
2. For #12, not correct, looks like he didn't understand the rules of the game. Algorithm is wrong. Also, pay attention to unbalanced curly braces which throw exceptions.
3. For #13, not sure if polar.plot is a proper function. Returned Error: could not find. Pay close attention to functions and function parameters.

Sarker (top)

1. For #7, cumbersome but correct.
2. In general, Sarker uses descriptive variable names and ensures the reader's understanding of the code.
3. With respect to functionality, the code does not produce any exceptions when executed and all results are logically correct.
4. Overall, this is a strong submission.

Shahinur Alam (top)

1. For #12, almost correct except for a small misunderstanding of the game. Once you get your point, say, 10, you ONLY "lose" by rolling a 7 (not 7 or 11), otherwise you continue rolling.
2. I would advise Shahinur to be sure to understand the algorithm(game) that he/she is trying to implement because if you don't understand the methods, there is no way to write accurate code for implementation.
3. With respect to the rest of the code, he/she uses descriptive variable names and clear comments, which always enhance readability and understandability.
4. With respect to functionality, the code does not produce any exceptions when executed and all other results are logically correct.
5. Overall, this is a fair submission.