BIOS545R Introduction to R Programming
2016 Spring Semester Final Exam

02/29/2016 3:00 – 4:50 PM

INSTRUCTIONS:

Answer all three questions. All work and code must be your own. Save all code into a single text file called YOURNAME_FINAL.R where YOURNAME should be replaced by your Last and First name. No email, text, chat (electronic or person-to-person) is allowed during the Final. You may not download packages to solve these problems. You may not Google for code. All code must be your own. Before leaving the room please email your file to dvandom@emory.edu, hao.wu@emory.edu and wsp@emory.edu.

Total is 100 points. Partial credit will be given.

QUESTION #1) (40 points)

Write a function called **numgen** that, given an input vector of two numbers, (called "range"), will compute a vector containing all odd numbers between the two numbers. If either of the beginning or end range values are themselves odd then include them in the return vector. You will also determine the sum of the computed vector and create a return list with two elements named **oddvec** and **sum** to contain the computed vector and sum**.**

It is easy to write this function using material covered in class as well as the Recaps on the website. You might use the R help facility to look for a built-in function that generates a vector/sequence of numbers after which you can do some testing to see which elements are odd (or not).

```
> numgen(c(2,11))
$oddvec
[1]  3  5  7  9 11

$sum
[1] 35

> numgen(c(3,12))
$oddvec
[1]  3  5  7  9 11

$sum
[1] 35
```

```
numgen <- function(range=c(2,11)) {

# INPUT: range – a vector specifying the begin and end of a range
# OUTPUT: a list containing 1) a vector element name "vec"
#         that has all odd numbers between the specified range.
#         2) an element name "sum" that contains the sum of the
#         "vec" element
#
#         If either or both the beginning or ending range numbers
#         are odd then return them also

  return(retlist)
}
```

QUESTION #2) (35 points)

When we use the mean function in R to compute the mean of a numeric vector x, we are typically wanting back what is known as the arithmetic mean which is the sum of all the elements in x divided by the length of the vector. That's easy. But there are two other 'means' (more than that actually): **1)** There is also the **harmonic mean**, which is computed as follows where *n* is the number of elements in the input vector and the *x* values represent the individual elements of the input vector.

$$H = \frac{n}{\frac{1}{x_1}+\frac{1}{x_2}+\frac{1}{x_3}+\frac{1}{x_n}} = \frac{n}{\sum_{i=1}^{n}\frac{1}{x_i}}$$

As an example here is how we would compute the harmonic mean of the numbers 1,3,4 which would be 1.894737

$$H = \frac{3}{\frac{1}{1}+\frac{1}{3}+\frac{1}{4}} = \frac{3}{\frac{12+4+3}{12}} = \frac{36}{19}$$

2) And there is the **geometric mean** that is computed as follows. *n* is the total number of elements in the input vector and *x* represents the values of the input vector. Below we take the product of all vector elements after which we raise it to an exponent, which is the reciprocal of n. So to compute the geometric mean of 1,3,4 we do the following to get a value of 2.289428

$$\left(\prod_{i=1}^{n} x_i\right)^{1/n} = (x_1 * x_2 * x_3 \ldots * x_n)^{\frac{1}{n}}$$

$$\left(\prod_{i=1}^{n} x_i\right)^{1/n} = (1 * 3 * 4)^{\frac{1}{3}}$$

Write a function called **mymean** that takes a numeric vector as input as well as a character string which denotes the type of mean to return: "arithmetic", "geometric", or "harmonic". You may NOT use R functions designed specifically to compute the mean (such as "mean") or download packages that compute the harmonic or geometric mean. However you can use various arithmetic functions (e.g. sum, prod, etc to help). Here are some examples:

```
> mymean(c(1,3,4),"arithmetic")
[1] 2.666667

> mymean(c(1,3,4),"harmonic")
[1] 1.894737

> mymean(c(1,3,4),"geometric")
[1] 2.289428

> mymean(1:20,"geometric")
[1] 8.304361

mymean <- function(x, type="harmonic") {

  # INPUT: x – a numeric vector
  #        type – a string denoting the desired mean type. "harmonic"
  #               is the default
  #
  # OUTPUT: a single value representing the desired mean

  return(somemean)

}
```

<u>QUESTION #3 (25 points)</u>

Consider the function *f(x)* defined as follows. This type of function is known as a "piecemeal" function. Note that the **log** function below represents the natural logarithm of x.

$$f(x) = \begin{cases} x^2, & x \le 0 \\ e^x, & 0 < x \le 1 \\ \log(x), & x > 1 \end{cases}$$

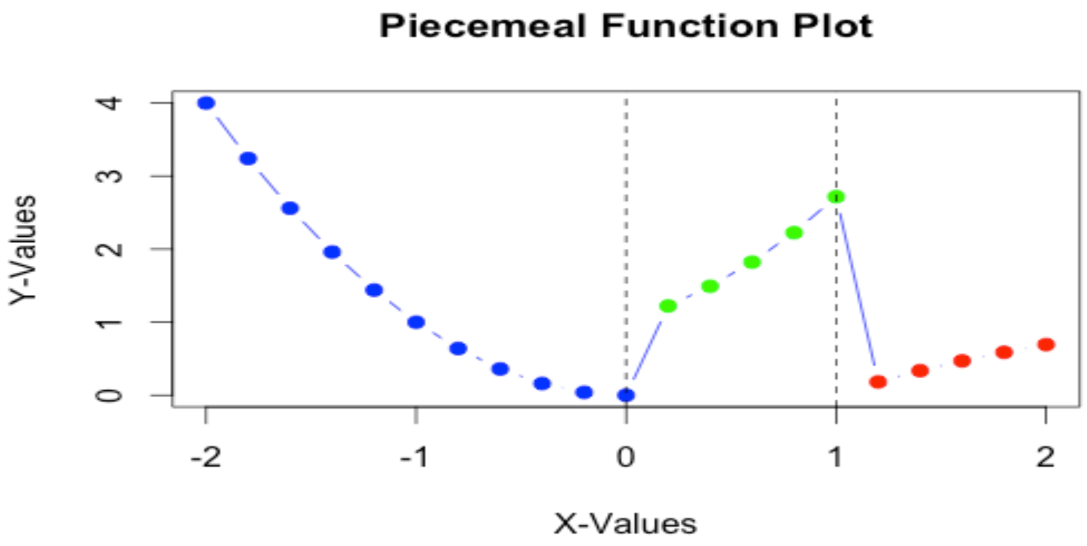
Create a function in R that implements the piecemeal definition. The function will take the following input: **xvals**: a single number or a vector for *x*.

The output of the function is **1)** a vector representing the *f(x)* values. The length of the output should be equal to the length of the input. **2)** a scatter plot of *f(x)* versus *x* will be shown using points and lines. The plot will include vertical dashed lines at x values 0 and 1.

The points <= 0 should be represented in the color blue. The points in the range 0 < x <= 1 should be represented in the color green. Points > 1 should be represented in the color red. As an example, you should get the following results given the following inputs. Note that your plots should have the same look as below including titles and axis labels.

```
myfunc( seq(-2, 2, 0.2) )
```

```
 [1] 4.0000000 3.2400000 2.5600000 1.9600000 1.4400000 1.0000000
 [7] 0.6400000 0.3600000 0.1600000 0.0400000 0.0000000 1.2214028
[13] 1.4918247 1.8221188 2.2255409 2.7182818 0.1823216 0.3364722
[19] 0.4700036 0.5877867 0.6931472
```



```
myfunc <- function(x) {

# INPUT:   A numeric vector containing a sequence of numbers
#
# OUTPUT: A vector of values representing f(x)
#
#          A plot like that above. Note that you don't
#          actually return the plot – you just execute the
#          plot from within this function

   return(fxvals)
}
```