STAT 2450 Assignment 3 (40 points)

Alice Liu

Banner: B00783546

1. Write a function to calculate miles per gallon given kilometres travelled, and litres of gasonline used. The function should have two arguments, litres and kilometres, and should return the mileage in mpg.

```r
mpg=function(litres,kilometres){#mpg=function(  arguments here){
   miles=kilometres*5/8#mpg=
   gallons=litres/4.55
   mpg=miles/gallons
   return(mpg) #return(mpg)
}
mpg(litres=8.3,kilometres=100)
```

```
## [1] 34.26205
```

Test your function using input values of 100 kilometres and 8.3 litres.

(5 points)

2. The roots of the quadratic $ax^2 + bx + c$ are given by

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

If $b^2 - 4ac < 0$, the quadratic has no real roots.

Write a function to calculate the real roots of a quadratic. The function should have 3 arguments, a, b and c. If $b^2 - 4ac < 0$, the function should print "quadratic has no real roots", and then return(NULL). Otherwise, the function should return a vector of length 2, those being the real roots (which may be the same if $b^2 - 4ac = 0$).

Test your function using the quadratic $x^2 - 3x + 2$.

```r
root=function(a,b,c){
   d=b^2-4*a*c
   if(d<0){
   print("quadratic has no real roots")
   return(NULL)
   }
   root=(-b+c(-1,1)*sqrt(d))/(2*a)
   return(root)
}
root(1,-3,2)
```

```
## [1] 1 2
```

(5 points)

3. Where $x_1, x_2, \ldots, x_n$ is a sample from a normal distribution with unknown mean μ and unknown variance σ^2, the level $100(1-\alpha)\%$ confidence interval for μ is given by

$$\bar{x} \pm t_{1-\alpha/2,n-1} \frac{s}{\sqrt{n}}$$

where $\bar{x}$ and s are the sample mean and sample standard deviation of the data, and $t_{1-\alpha/2,n-1}$ cuts off an area $1-\alpha/2$ to its left under the t curve with $n-1$ degrees of freedom.

Write a function which has two arguments, a vector of data x, and alpha, which should have a default value of .05. The function should return a vector of length 2, which contains the endpoints of the confidence interval.

The percentiles of the t-distrubtion can be calculated as follows. Suppose that you want the 97.5'th percentile of the t-distribution with 23 degrees of freedom. This can be calculated in R as

```r
qt(.975,23)
```

```
## [1] 2.068658
```

Test your function by calculating the 99% confidence interval using the following data

```r
set.seed(87612345)
data=rnorm(25,mean=4.5,sd=.75)
```

You can check your calculation using

```r
t.test(data,conf.level=.99)
```

```
##
##  One Sample t-test
##
## data:  data
## t = 29.832, df = 24, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 99 percent confidence interval:
##  4.109001 4.959191
## sample estimates:
## mean of x
##  4.534096
```

When putting your two endpoints together, you may find something similar to the following to be useful.

```r
1+c(-1,1)*.25
```

```
## [1] 0.75 1.25
```

```
t_interval=function(data,alpha=.05){
    n=length(data)
    t=qt(1-alpha/2,n-1)
    xbar=mean(data)
    s=sd(data)
    t_interval=xbar+c(-1,1)*t*s/sqrt(n)
return(t_interval)
}
t_interval(data,.01)
```

```
## [1] 4.109001 4.959191
```

(5 points)

4. The derivative of a function $f(x)$ can be approximated by the Newton's quotient

$$\frac{f(x+h)-f(x)}{h}$$

where h is a small number. Write a function to calculate the Newton's quotient for $f(x) = exp(x)$. The function should take two scalar arguments, x and h. Use a default value of $h = 1.e-6$. Test your function at the point $x = 1$ using the default value of h, and compare to the true value of the derivative $f'(1) = e^1$.

```
newton_equation=function(x,h=1e-6){
    temp=(exp(x+h)-exp(x))/h
    return(temp)
}
newton_equation(1)
```

```
## [1] 2.718283
```

(5 points)

5. A very useful feature in R is the ability to pass a function name as an argument. Here is an example, where 2 is added to the value of a function, for three different functions $exp(x)$, $log(x)$, and $sin(x)$, at selected points x.

```
test=function(x,f){
 output=f(x)+2
 return(output)
  }

test(0,exp)
```

```
## [1] 3
```

```
test(1,log)
```

```
## [1] 2
```

```r
test(0,sin)
```

```
## [1] 2
```

```r
test(pi/2,sin)
```

```
## [1] 3
```

Modify your function from problem 4 so that you pass in the name of the function for which you want to approximate the derivative. Use the same default value for h, and approximate the derivative of sin(x) at $x = \pi/4$, of $log(x)$ at $x = 2$, and of $exp(x)$ at $x = 1$.

```r
#h is the default value,
#and should be put in the last. Otherwise it doesn't know which one is default value.
newtonf2=function(x,f,h=1e-6){
  temp=(f(x+h)-f(x))/h
  return(temp)
}
newtonf2(pi/4,sin)
```

```
## [1] 0.7071064
```

```r
newtonf2(2,log)
```

```
## [1] 0.4999999
```

```r
newtonf2(1,exp)
```

```
## [1] 2.718283
```

(10 points)

6. Write a function which takes one argument x of length 2, and returns the ordered values of x. That is, if $x_2 < x_1$, your function should return $c(x_2, x_1)$, otherwise it should return x. (WRITE YOUR OWN FUNCTION. DO NOT USE THE BUILT IN FUNCTION ORDER)

Use your function to process a dataset with 2 columns as follows. Iterate over the rows of the data set, and if the element in the 2nd column of row i is less than the element in the first column of row i, switch the order of the two entries in the row by making a suitable call to the function you just wrote.

Test using the following data.

```r
set.seed(1128719)
data=matrix(rnorm(20),byrow=T,ncol=2)
data
```

```
##                [,1]       [,2]
##  [1,] -0.04142965  0.2377140
##  [2,] -0.76237866 -0.8004284
##  [3,]  0.18700893 -0.6800310
##  [4,]  0.76499646  0.4430643
##  [5,]  0.09193440 -0.2592316
##  [6,]  1.17478053 -0.4044760
##  [7,] -1.62262500  0.1652850
##  [8,] -1.54848857  0.7475451
##  [9,] -0.05907252 -0.8324074
## [10,] -1.11064318 -0.1148806
```

```r
sortf=function(x){
  if(x[2]<x[1]){
  x=c(x[2],x[1])
}
return(x)
}

sortf2=function(x){
n=nrow(x)
for (i in 1:n){
x[i,]=sortf(x[i,])
}
return(x)
}
sortf2(data)
```

```
##                [,1]        [,2]
##  [1,] -0.04142965  0.23771403
##  [2,] -0.80042842 -0.76237866
##  [3,] -0.68003104  0.18700893
##  [4,]  0.44306433  0.76499646
##  [5,] -0.25923164  0.09193440
##  [6,] -0.40447603  1.17478053
##  [7,] -1.62262500  0.16528496
##  [8,] -1.54848857  0.74754509
##  [9,] -0.83240742 -0.05907252
## [10,] -1.11064318 -0.11488062
```

(10 points)