Problem 1

1. You need to read `chol.txt` file for this problem.

   - Calculate the overall mean for each variable, except for the `sex` variable.

   ```
   > chol=read.table("chol.txt")
   > apply(chol[,-1], 2, mean, na.rm=T)
   ```

   ```
         age       chol         tg         ht         wt        sbp        dbp
   22.656250 198.568421  80.373684  61.428490 123.551042 120.947917  74.567708
        vldl        hdl        ldl        bmi
   15.426316  48.678947 131.684211   3.100473
   ```

   - Calculate the sex-specific mean for each variable.

   ```
   > aggregate(chol[,-1], list(chol$sex), mean, na.rm=T)
   ```

   ```
     Group.1      age     chol       tg       ht       wt      sbp      dbp
   1       F 22.88542 202.2000 79.45263 59.19719 109.5104 117.1250 73.31250
   2       M 22.42708 194.9368 81.29474 63.65979 137.5917 124.7708 75.82292
         vldl      hdl      ldl      bmi
   1 15.23158 51.40000 132.1158 2.977679
   2 15.62105 45.95789 131.2526 3.223266
   ```

2. Create a new variable, `chol2`, which is based on the `chol` variable from the `chol` data frame. If `chol` is greater than the mean of `chol`, then `chol2 = 'Hi'`; otherwise, `chol2= 'LOW'`

   - Calculate the standard deviation of `bmi` for each `chol2` category.

   ```
   > chol2 = rep("LOW", nrow(chol))
   > chol2[chol$chol > mean(chol$chol, na.rm=T)]= "HI"
   > chol2[is.na(chol$chol)]= NA
   > tapply(chol$bmi, chol2, sd)
   ```

   ```
         HI      LOW
   0.787943 0.683698
   ```

   - Calculate the standard deviation of `bmi` for each combination of `sex` and `chol2` categories.

   ```
   > tapply(chol$bmi, list(chol$sex, chol2), sd)
   ```

   ```
            HI       LOW
   F 0.7304674 0.7392027
   M 0.7847937 0.6258757
   ```

Problem 2

Consider the simulated matrix, mat, with 10 rows and 20 columns.

```
> set.seed(91765)
> mat = matrix(rnorm(200), 10)
> mat[1,1] = NA
> dim(mat)
```

```
[1] 10 20
```

1. Calculate the median (use the `median` function) of each row by using the `for` loop

```
> row.median = numeric(10)  ## allocate space for the result
> for (i in 1:10){
+    row.median[i] = median(mat[i,], na.rm=T)
+ }
> row.median

 [1]  0.099710574  0.937729599 -0.331428032  0.014436358  0.006769759
 [6] -0.271493036  0.223632804 -0.233758825 -0.012546087 -0.249463035
```

2. Calculate the median of each row by using the `apply` function

```
> apply(mat, 1, median, na.rm=T)

 [1]  0.099710574  0.937729599 -0.331428032  0.014436358  0.006769759
 [6] -0.271493036  0.223632804 -0.233758825 -0.012546087 -0.249463035
```

Problem 3

When performing a two-sample t-test, the data is organized into columns. However, sometimes data is organized into rows, such as genetic data. Each row contains values for each gene, and you would like to perform a two-sample t-test for each gene.

Write a function, `rowTtest`, to perform a row-wise two-sample t-test. This function will take two main arguments: the first one is a matrix or data frame with each row representing values for each gene, and with the number of columns equaling the sample size. The second argument is a character vector with length equaling sample size containing the phenotype information. The returned value is the input data frame with only those rows (genes) being statistically significant ($p < (0.05/\#$ of Genes), based on `Bonforoni` correction).

To test this function, you need to simulate a data frame or a matrix with `dim = c(1000, 20)`. Make sure to generate row names, from `G1 - G1000`. The first 10 columns are the cases and the second 10 columns are the controls. First you will start to simulate each value following standard normal distribution. Then add 4 to G1 - G10 for only the cases. Test if you will be able to find the first 10 genes based on the function that you created. Note, you may not get those exact 10 genes, but it should be close.

Please do not use the iterative DO-loop (such as `for`, `while`, or `repeat`) inside the function.

Here are some hint/suggestions for problem 3:

You need to simulate a matrix with dimension of 1000 by 20. For example, this matrix will look like the following:

```
> set.seed(1)
> gene = matrix(rnorm(1000*20), 1000)
> gene[1:10, 1:10] <- gene[1:10, 1:10]+4
> rownames(gene) <- paste("G", 1:1000, sep="")

> dim(gene)

[1] 1000   20
```

```
> head(gene)

       [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
G1 3.373546 5.134965 3.113850 4.739115 2.865370 2.483627 3.381173 2.674582
G2 4.183643 5.111932 2.077745 4.386609 4.764557 4.629141 2.890578 4.951980
G3 3.164371 3.129222 5.619701 5.296397 4.570710 2.321806 1.829665 4.860004
G4 5.595281 4.210732 4.519270 3.196442 2.648306 5.179781 3.968697 5.060790
G5 4.329508 4.069396 3.944150 2.397374 1.970115 5.117655 3.739602 3.649416
G6 3.179532 2.337351 4.696418 4.933251 4.590479 2.762264 4.534430 3.869234
       [,9]    [,10]      [,11]      [,12]       [,13]      [,14]      [,15]
G1 4.263703 2.782880 -0.8043316 -1.4115219 -0.93910663  0.2264537  0.5232667
G2 3.170548 3.053771 -1.0565257  1.0838697  1.39366493 -0.8185942  0.9935537
G3 2.538365 4.091410 -1.0353958  1.1702224  1.62581486 -0.8471526  0.2737370
G4 5.683990 4.701351 -1.1855604  0.2947545  0.40900106 -1.9843326 -0.6949193
G5 2.455676 4.673422 -0.5004395 -0.5544277 -0.09255856 -0.8127788 -0.7180502
G6 3.809113 5.265553 -0.5249887 -0.4034407  0.20609871  1.4616707 -0.1019895
        [,16]      [,17]      [,18]      [,19]       [,20]
G1 -0.2139090  0.8576341  1.0496171  0.9514099 -2.07771241
G2 -0.1067233 -1.6253951  0.2903237  0.4570987 -0.45446091
G3 -0.4645893 -0.2342783  1.2421262 -0.3586935 -0.16555991
G4 -0.6842725 -1.0326545 -0.6850857 -1.0458614  0.89765209
G5 -0.7908007 -1.1411412 -0.6677681  0.3075345 -0.02948916
G6 -0.3389638 -1.5219369  0.9409138  1.9943876  1.85838843
```

Note that your simulated data will probably have different numbers. You need to provide the matrix `gene` above as the first argument of your function. You also need to create a character vector like the one below:

```
> pheno = c(rep("case", 10), rep("control", 10))

> pheno

 [1] "case"    "case"    "case"    "case"    "case"    "case"    "case"
 [8] "case"    "case"    "case"    "control" "control" "control" "control"
[15] "control" "control" "control" "control" "control" "control"

> length(pheno)

[1] 20
```

Within the function, you need to perform a t-test for each gene. For example, to get the p-value for the first gene, you can write the following:

```
> t.test(gene[1,] ~ pheno)$p.value

[1] 2.292288e-07
```

Similarly, to obtain the p-value for the 100th gene, you can do the following:

```
> t.test(gene[100,] ~ pheno)$p.value
```

```
[1] 0.2998438
```

In your function, you need to perform 1000 t-test and grab the p-value from each test and store these 1000 p-value to vector, named `p`. Generally we use 0.05 as a significant cutoff value. Since we are performing 1000 t-test, we need to use a Bonforoni adjusted p-value (0.05/1000). The returned value of the function is the input data frame or matrix with only those rows being significant.

```
> rowTtest = function(data, phenoInfo, ...){
+     p = apply(data, 1, function(y){
+         t.test(y ~ phenoInfo, ...)$p.value
+     })
+     r = data[p <(0.05/nrow(data)),]
+     return(r)
+ }
> rowTtest(gene, pheno)
```

```
        [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
G1  3.373546 5.134965 3.113850 4.739115 2.865370 2.483627 3.381173 2.674582
G2  4.183643 5.111932 2.077745 4.386609 4.764557 4.629141 2.890578 4.951980
G3  3.164371 3.129222 5.619701 5.296397 4.570710 2.321806 1.829665 4.860004
G4  5.595281 4.210732 4.519270 3.196442 2.648306 5.179781 3.968697 5.060790
G5  4.329508 4.069396 3.944150 2.397374 1.970115 5.117655 3.739602 3.649416
G6  3.179532 2.337351 4.696418 4.933251 4.590479 2.762264 4.534430 3.869234
G7  4.487429 4.810840 4.053516 5.806089 2.586930 2.769835 3.440561 4.763586
G8  4.738325 2.087654 2.689717 3.943496 5.610342 4.597791 5.608370 3.506094
G9  4.575781 2.753247 1.876934 5.885911 5.840443 4.298864 4.556640 5.113360
G10 3.694612 4.998154 3.791921 5.578383 5.368298 3.889861 4.185622 5.458963
        [,9]    [,10]      [,11]        [,12]       [,13]      [,14]      [,15]
G1  4.263703 2.782880 -0.8043316 -1.411521883 -0.93910663  0.2264537  0.5232667
G2  3.170548 3.053771 -1.0565257  1.083869657  1.39366493 -0.8185942  0.9935537
G3  2.538365 4.091410 -1.0353958  1.170222351  1.62581486 -0.8471526  0.2737370
G4  5.683990 4.701351 -1.1855604  0.294754540  0.40900106 -1.9843326 -0.6949193
G5  2.455676 4.673422 -0.5004395 -0.554427663 -0.09255856 -0.8127788 -0.7180502
G6  3.809113 5.265553 -0.5249887 -0.403440689  0.20609871  1.4616707 -0.1019895
G7  5.016212 2.554978 -0.3024330 -1.268123219 -0.09160289  0.6132922  0.2122542
G8  4.547126 5.415418  0.4719681 -0.009138438 -0.46000136  0.2994154  1.6316872
G9  4.755154 2.414966 -0.2483839  0.341759323  0.56749428 -0.4803364  0.2547482
G10 3.580196 4.245757  1.2593180  0.394039668 -1.46018697  2.4568363 -2.0730307
         [,16]       [,17]      [,18]      [,19]       [,20]
G1  -0.2139090  0.85763410  1.0496171  0.9514099 -2.07771241
G2  -0.1067233 -1.62539515  0.2903237  0.4570987 -0.45446091
G3  -0.4645893 -0.23427831  1.2421262 -0.3586935 -0.16555991
G4  -0.6842725 -1.03265445 -0.6850857 -1.0458614  0.89765209
G5  -0.7908007 -1.14114122 -0.6677681  0.3075345 -0.02948916
G6  -0.3389638 -1.52193690  0.9409138  1.9943876  1.85838843
G7  -1.2674299 -0.75489194  0.9953313 -1.8302888 -1.20876167
G8  -1.3941059 -0.46676191  1.0210044  1.3443111 -1.70210829
G9   0.4189249  0.05908485  0.7925958 -1.4806929 -0.81429564
G10  3.2151854  1.41329824 -1.3772293 -0.7172221  1.27011658
```

Problem 4

Calculate the two-sample t-test for each numerical variable in the `chol` data frame (from the `chol.txt` file) by the `sex` variable by using the `apply` function. The result should be a matrix that contains five columns: F.mean (mean of the numerical variable for Female), M.mean (mean of the numerical variable for Male), t (for t-statistics), df (for degrees of freedom), and p (for p-value). The result should look like the one below:

```
> result = t(apply(chol[,-1], 2, function(x){
+       foo =  t.test(x ~ chol$sex)
+       c(F.mean = as.vector(foo$estimate[1]),
+         M.mean = as.vector(foo$estimate[2]),
+         t = as.vector(foo$statistic),
+         df = as.vector(foo$parameter),
+         p =foo$p.value)
+ }))

> result

          F.mean     M.mean           t       df            p
age   22.885417  22.427083  0.20489826 187.9970 8.378733e-01
chol 202.200000 194.936842  0.75035027 188.0000 4.539818e-01
tg    79.452632  81.294737 -0.28380022 158.5931 7.769337e-01
ht    59.197187  63.659792 -3.42175696 186.0388 7.649414e-04
wt   109.510417 137.591667 -4.09786339 183.3389 6.254578e-05
sbp  117.125000 124.770833 -0.99609766 113.1632 3.213279e-01
dbp   73.312500  75.822917 -1.32775215 189.0118 1.858613e-01
vldl  15.231579  15.621053 -0.30479390 164.7068 7.609079e-01
hdl   51.400000  45.957895  3.37529526 187.9173 8.958234e-04
ldl  132.115789 131.252632  0.09293561 187.5109 9.260539e-01
bmi    2.977679   3.223266 -2.39434622 189.4748 1.762516e-02
```