ACADGILD ASSIGNMENT 3.4

A) Implement user defined functions within apply function using the mtcars data set and produce column wise summary statistics using apply function and mtcars dataset.

ANSWER : The apply() functions form the basis of more complex combinations and helps to perform operations with very few lines of code..

We will use mtcars dataset. Let's start with adding labels to the dataset. Then we can continue with tables creation.

Example 1 :

```
# show first few rows of mtcars
head(mtcars)
```

```
 model     mpg  cyl  disp   hp  drat   wt  qsec   vs   am  gear  carb
 <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 Mazda R~  21     6   160   110  3.9   2.62  16.5    0    1    4    4
 2 Mazda R~  21     6   160   110  3.9   2.88  17.0    0    1    4    4
 3 Datsun ~  22.8   4   108    93  3.85  2.32  18.6    1    1    4    1
 4 Hornet ~  21.4   6   258   110  3.08  3.22  19.4    1    0    3    1
 5 Hornet ~  18.7   8   360   175  3.15  3.44  17.0    0    0    3    2
 6 Valiant   18.1   6   225   105  2.76  3.46  20.2    1    0    3    1
```

```
>
# get the mean of each column
apply(mtcars, 2, mean)
 mpg       cyl      disp        hp      drat        wt
 20.090625  6.187500 230.721875 146.687500  3.596563  3.217250
 qsec        vs        am      gear      carb
 17.848750  0.437500  0.406250  3.687500  2.812500

# get the sum of each row (not really relevant for this data
# but it illustrates the capability)
apply(mtcars, 1, sum)
Mazda RX4       Mazda RX4 Wag         Datsun 710
328.980           329.795           259.580
Hornet 4 Drive    Hornet Sportabout           Valiant
426.135           590.310           385.540
Duster 360          Merc 240D           Merc 230
656.920           270.980           299.570
Merc 280          Merc 280C          Merc 450SE
350.460           349.660           510.740
Merc 450SL        Merc 450SLC   Cadillac Fleetwood
511.500           509.850           728.560
Lincoln Continental   Chrysler Imperial            Fiat 128
726.644           725.695           213.850
```

```
Honda Civic      Toyota Corolla       Toyota Corona
195.165          206.955              273.775
Dodge Challenger         AMC Javelin          Camaro Z28
519.650          506.085              646.280
Pontiac Firebird         Fiat X1-9       Porsche 914-2
631.175          208.215              272.570
Lotus Europa      Ford Pantera L       Ferrari Dino
273.683          670.690              379.590
Maserati Bora         Volvo 142E
694.710          288.890

# get column quantiles (notice the quantile percents as row names)
apply(mtcars, 2, quantile, probs = c(0.10, 0.25, 0.50, 0.75, 0.90))
     mpg cyl    disp    hp  drat      wt    qsec vs am gear carb
10% 14.340   4  80.610  66.0 3.007 1.95550 15.5340  0  0    3    1
25% 15.425   4 120.825  96.5 3.080 2.58125 16.8925  0  0    3    2
50% 19.200   6 196.300 123.0 3.695 3.32500 17.7100  0  0    4    2
75% 22.800   8 326.000 180.0 3.920 3.61000 18.9000  1  1    4    4
90% 30.090   8 396.000 243.5 4.209 4.04750 19.9900  1  1    5    4
```

B) Write a program to extract the names of the list.

ANSWER :

Example 1: A list is an R structure that may contain object of any other types, including other lists. Lots of the modeling functions (like t.test() for the t test or lm() for linear models) produce lists as their return values, but you can also construct one yourself:

```
install.packages("rlist")
```

```
library(rlist)
devs <-
list(
p1=list(name="Ken",age=24,
interest=c("reading","music","movies"),
lang=list(r=2,csharp=4)),
p2=list(name="James",age=25,
interest=c("sports","music"),
lang=list(r=3,java=2,cpp=5)),
p3=list(name="Penny",age=24,
interest=c("movies","reading"),
lang=list(r=1,cpp=4,python=2)))
```

Examples use str()

Step 2 : Filter those who like music and has been using R for more than 3 years.

```
str( list.filter(devs, "music" %in% interest & lang$r >= 3) )
```

```
 List of 1
 $ p2:List of 4
..$ name    : chr "James"
..$ age     : num 25
..$ interest: chr [1:2] "sports" "music"
..$ lang    :List of 3
..$ r   : num 3
..$ java: num 2
..$ cpp : num 5
```

Step 3: Selecting

```
str( list.select(devs, name, age) )
```

```
List of 3
 $ p1:List of 2
..$ name: chr "Ken"
..$ age : num 24
 $ p2:List of 2
..$ name: chr "James"
..$ age : num 25
 $ p3:List of 2
..$ name: chr "Penny"
..$ age : num 24
```

Step 4: Mapping

```
str( list.map(devs, length(interest)) )
```

```
List of 3
$ p1: int 3
$ p2: int 2
$ p3: int 2
```