Practice Final Exam

You'll find below the practice exam from last semester. The format of the exam will be different, since you will be doing the exam on real computer and not on a sheet of paper, but trying it will nevertheless be a very good exercise to help you prepare for the final.

In [130]:
```
%load_ext rmagic
%R rm()
```

```
The rmagic extension is already loaded. To reload it, use:
  %reload_ext rmagic
```

PART I: Vector and Function Basics

In the cell below, write **vectorized code** (i.e. no for loops or if statements) to create the following vectors:

- $(1, 2, 3, \ldots, 19, 20, 19, 18, \ldots, 2, 1)$

In [131]:
```
%%R
x = c(1:19,20:1); print(x)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 19 18 17
16 15
[26] 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

- $(4, 4, \ldots, 4, 6, 6, \ldots, 6, 3, 3, \ldots, 3)$ where 4 appears 10 times, 6 appears 20 times, and 3 appears 30 times

In [132]:
```
%%R
x = rep(c(4,6,3), c(10, 20, 30)); print(x)
```

```
 [1] 4 4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 3 3 3 3 3
 3 3 3
[39] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

- ('label 1', 'label 2',..., 'label 30')

In [133]:
```
%%R

labels = paste('label', 1:30); print(labels)
```

```
 [1] "label 1"  "label 2"  "label 3"  "label 4"  "label 5"  "label 6"
 [7] "label 7"  "label 8"  "label 9"  "label 10" "label 11" "label 12"
[13] "label 13" "label 14" "label 15" "label 16" "label 17" "label 18"
[19] "label 19" "label 20" "label 21" "label 22" "label 23" "label 24"
[25] "label 25" "label 26" "label 27" "label 28" "label 29" "label 30"
```

- Create a vector containing the values $e^x \cos(x)$ at $x = 3, 3.1, 3.2, \ldots, 6$.

In [134]:
```
%%R
x = seq(3, 6, by=0.1)
y = exp(x)*cos(x); print(y)
```

```
 [1] -19.884531 -22.178753 -24.490697 -26.773182 -28.969238 -31.011186
 [7] -32.819775 -34.303360 -35.357194 -35.862834 -35.687732 -34.685042
[13] -32.693695 -29.538816 -25.032529 -18.975233 -11.157417  -1.362099
[19]  10.632038  25.046705  42.099201  61.996630  84.929067 111.061586
[25] 140.525075 173.405776 209.733494 249.468441 292.486707 338.564378
[31] 387.360340
```

- Create the vector $x = (1/2^1, \ldots, 1/2^{200})$ and then calculate the following sum in a vectorized way

$$\sum_{i=1}^{199} \frac{e^{-x_{i+1}}}{x_i + 10}$$

In [135]:
```
%%R

x = 1/2^(1:200)
i = 1:199
result = sum(exp(-x[i+1])/(x[i] + 10)); print(result)
```

```
[1] 19.84565
```

- Write a function `simulate.weatherFrame(n, T.freq, H.freq)` that return a randomly generated data frame with n observations of the two categorical variables: `Temperature` whose values can "Cold", "Warm", and "Hot" and `Humidity` whose values can be "Low", "Normal", and "High". The `T.freq` argument should specify the occurence frequencies of "Cold", "Warm", and "Hot", while the `H.freq` argument should specify the occurence frequencies of "Low", "Normal", and "High". The type of both variables should be `factor`.

In [175]:
```
%%R

simulate.weatherFrame = function(n, T.freq, H.freq)
{
    Temp = factor(sample(c('Cold', 'Warm', 'Hot')  , n, replace=T, prob=T
.freq))
    Humi = factor(sample(c('Low', 'Normal', 'High'), n, replace=T, prob=H
.freq))
    df = data.frame(Temperature=Temp, Humidity=Humi)
    return(df)
}
```

In [177]:
```
%%R
```

```
df = simulate.weatherFrame(200, c(0.2, 0.3, 0.5), c(0.2, 0.3, 0.5))
print(head(df))
```

```
  Temperature Humidity
1        Warm   Normal
2        Warm   Normal
3        Cold     High
4         Hot   Normal
5        Warm   Normal
6         Hot     High
```

Part II: Writting and Using Classes

Consider the following character vector containing personal information:

- Write the constructor of a class `contact` that has the following attributes: `Name`, `Phone`, `Email`, and `Age`.

In [138]:
```
%%R

contact = function(name, phone, email, age)
{
    object = list(Name=name, Phone=phone, Email=email, Age=age)
    class(object) = 'contact'
    return(object)
}
```

In [139]:
```
%%R

bob = contact('Bob Duran', '324 5342', '234@ljlj.com', 12)
paul = contact('Paul Duran', '324 5342', '234@ljlj.com', 12)
jane = contact('Jane Duran', '324 5342', '234@ljlj.com', 12)

print(bob$Phone)
print(class(bob))
```

```
[1] "324 5342"
[1] "contact"
```

- Write the constructor of a class `addressBook` that has the following attributes: `owner`, which is a string containing the address book owner name, `contacts` which is meant to store a list of `contact` objects, and `fields` which is a character vector containing the contact fields (i.e. "Name", "Phone", and "Email").

The class constructor should set the contact list attribute to the empty list if no `contact` argument is passsed to it.

The class constructor should also create the attribute `fields` and store the contact fields. Because of that, there is no need of a `fields` argument for the class constructor.

In [140]:
```
%%R

addressBook = function(owner, contacts=list())
{
    object = list(owner=owner,
                  contacts=contacts,
                  fields=c('Name', 'Phone', 'Email', 'Age')
                 )
    class(object) = 'addressBook'
    return(object)
}
```

In [141]:
```
%%R

a = addressBook('Bertrant Blue', list(bob, paul, jane))
```

- Write a method `as.data.frame` for the class `addressBook` that returns a data frame of the contacts in the address book. (The column should have labels 'Name', 'Phone', 'Email', and 'Age' and be of the appropriate column types.)

You should use the apply function family here and vectorized code. If you can not do it this way, try with for loops, but points will be deducted.

In [142]:
```
%%R

as.data.frame.addressBook = function(object)
{
    getField = function(field)
    {
        sapply(object$contacts, function(contact) contact[[field]]
    )
    df = as.data.frame(lapply(object$fields, getField))
    colnames(df) = object$fields
    return(df)
}
```

In [143]:
```
%%R

df= as.data.frame(a)
print(df)
```

```
        Name    Phone        Email Age
1  Bob Duran 324 5342 234@ljlj.com  12
2 Paul Duran 324 5342 234@ljlj.com  12
3 Jane Duran 324 5342 234@ljlj.com  12
```

- Using the method you just created, write a `print` method for the class `addressBook` that displays the name of the address book owner followed by the contact information displayed as a data table:

In [144]:
```
%%R
```

In [144]:
```
%%R

print.addressBook = function(object) print(as.data.frame(object))
```

In [145]:
```
%%R

print(a)
```

```
       Name    Phone        Email Age
1  Bob Duran 324 5342 234@ljlj.com  12
2 Paul Duran 324 5342 234@ljlj.com  12
3 Jane Duran 324 5342 234@ljlj.com  12
```

- Suppose our contact data are gathered in the form of character vectors as below:

In [146]:
```
%%R

info = c('Bob Durant 56years bob@bibishop.com',
         'Dan Bribri 10years 415 7838',
         'Brig Farty 38years 510 3478',
         'Cart Shuok 34years Cart@RATROU.com',
         'Krug Erbil 54years Cart@RATROU.com 510 230 450')
```

- Using regular expressions, write a function `create.addressBook(info, owner)` that takes a character vector as above and returns an `addressBook` object with the contact attribute set appropriately for each element of `info`. Write helper functions to break down the problemn and your code in small chunks: for instance, you may want to write getName, getPhone, etc. fucntions

In [172]:
```
%%R

getName  = function(info) sub('[0-9]{2}years.*', '', info)
getPhone = function(info) sub('^[^0-9 ]*', '', sub('^.*years *', '', info
))
getEmail = function(info) sub('[ 0-9]*$', '', sub('^.*years *', '', info)
)
getAge   = function(info) as.numeric(sub('years.*', '', sub('^[^0-9]+', '
', info)))

create.addressBook = function(info, owner)
{
    info.df = data.frame(getName(info),
                         getPhone(info),
                         getEmail(info),
                         getAge(info)
                        )
    contact.df = as.data.frame(t(info.df), stringsAsFactors=F)
    contact.list = lapply(contact.df,
                          function(X) contact(name=X[1],
                                              phone=X[2],
                                              email=X[3],
                                              age=X[4])
                                )
```

```
    return(addressBook(owner, contact.list))

}
```

In [173]:
```
%%R

clist = create.addressBook(info, 'Paul Rastaboom')
print(clist)
```

```
          Name        Phone             Email Age
V1 Bob Durant              bob@bibishop.com  56
V2 Dan Bribri      415 7838                   10
V3 Brig Farty      510 3478                   38
V4 Cart Shuok                Cart@RATROU.com  34
V5 Krug Erbil   510 230 450  Cart@RATROU.com  54
```

Part III: Refactoring Code

The following code simulates a data frame containing the homework, midterm, and final grades of a given class. The grades and the student names are randomly generated. This code is not optimal for two reasons:

- it is not broken into simple **functions**, which we may reuse to simulate data frame of the same type
- it is not **vectorized**, since it loops and branches over vector elements with `for` and `if` statements

In [35]:
```
%%R

student.number = 200

Homework = rnorm(student.number, mean=50, sd=30)
Midterm  = rnorm(student.number, mean=50, sd=30)
Final    = rnorm(student.number, mean=50, sd=30)
grades   = data.frame(F=Final, M=Midterm, H=Homework)

for(i in 1:nrow(grades)){
    for(j in 1:ncol(grades)){
        if(grades[i,j] > 100) grades[i,j] = 100
        if(grades[i,j] < 0  ) grades[i,j] = 0
    }
}


V = c('a', 'e', 'i','o','u','au','in','an','on','ou')
C = letters[-grep('a|e|i|o|u', letters)]
I = toupper(C)

simulated.names = c()

for (i in 1:student.number){
```

```
    first.name = paste(sample(I,1), sample(V,1), sample(C,1), sample(V,1),
 sep='')
    last.name  = paste(sample(I,1), sample(V,1), sample(C,1), sample(V,1),
 sep='')
    name       = paste(first.name, last.name, sep=' ')
    simulated.names = c(simulated.names, name)
}


row.names(grades) = simulated.names
```

Here is what the first entries in the simulated data frame `grades` look like:

In [36]:
```
%%R
print(head(grades))
```

```
                     F        M         H
Hujan Sinyon  26.11236 68.18481   0.00000
Cigon Lagau   14.68318 30.65834  39.24186
Naju Kanu     60.51722 87.85363 100.00000
Wongan Doube   0.00000 49.46895  37.94274
Dinli Qohu    50.92315 33.16281  59.40605
Ronnau Fonman  3.82837 87.45357  35.60847
```

The goal of this exercise is to **break this code into reusable functions and to vectorized it**. In the cell below, you'll break this code into the following functions:

- a function `simulate.names(n)` returning a character vector of n randomly generated names in the format above
- a function `clean.grades(df, min=0, max=100)` that takes a grade dataframe with any number of columns and rows and set the grades above `max` to `max` and the grades below `min` to `min`. The function should return the data frame with cleaned grades.
- a function `simulate.gradeFrame(col.names, student.number, grade.range)` that takes a character vector `col.names` containing the column names and returns a data frame with as many columns as elements in the first argument and as many rows as specified by the integer `student.number`. The grades should be between `grade.range[1]` and `grade.range[2]`, which is an numeric vector with two elements. The rows should be labelled with randomly generated student names. Use the function you wrote before in this function body.

In the last cell, use your `simulate.gradeFrame` function to simulate the same type of grade frame as above.

CAUTION: Your code should be **vectorized**. Points will be removed for any `for` loops and `if` statements your write: ideally, you should write note of theses. You can write as many helper functions as you wish, if you feel the need to.

In [37]:
```
%%R
```

In [37]:
```
%%R

chunks = function(A, n) sample(A, n, replace=T, prob=rep(1/length(A), leng
th(A)))

singleNames = function(n) paste(chunks(I,n),
                                chunks(V,n),
                                chunks(C,n),
                                chunks(V,n),
                                sep=''
                               )

simulate.names = function(n) paste(singleNames(n),
                                   singleNames(n),
                                   sep=' '
                                  )
```

In [38]:
```
%%R

clean.grades = function(df, min=0, max=100)
{
    df[df < min] = min
    df[df > max] = max
    return(df)
}
```

In [39]:
```
%%R

simulate.gradeFrame = function(col.names, student.number, grade.range)
{
    grades = sapply(1:length(col.names),
                    function(x) rnorm(student.number, mean=50, sd=30)
                    )

    grades = as.data.frame(clean.grades(grades))

    colnames(grades) = col.names
    rownames(grades) = simulate.names(student.number)
    return(grades)
}
```

In [40]:
```
%%R
grades = simulate.gradeFrame(c('F', 'M', 'H'), 200)
print(head(grades))
```

```
                   F        M        H
Zonfon Vija  36.65617 89.60199 75.28259
Hako Noyan   54.05560 67.66865 25.17915
Penon Kosu   72.88547 27.47065 71.28733
Vonzo Yeqau  28.38218 28.54643 71.88899
Minhau Kanca 80.24022 50.67674 66.28161
Cinzu Kinau  55.19146 30.61899 19.83443
```

In [40]:

In [40]:

In []: