1. **Write a R program using control operators to test whether following values are prime numbers or not by providing a PRIME or NOT PRIME message as output :**

Ans-

Primes <- function(x){

if (x == 2) {

print("PRIME")

} else {

if (any(x%% 2:(x-1) == 0)) {

print("PRIME")

} else {

print("NOT PRIME")

}}

1. **Write a R program using control operators to identify letter u and a both occur in the following words:**

car<- c("above","unit","Under")

for (i in car){

print(grep(i,c("u","a"),car,value=TRUE))

}

1. **Write a function that to calculate BMI (Body Mass Index):**

HW<-function(w,h)

{

if ((w/h\*h)<15) print("Very severely underweight")

if ((w/h\*h)>=15 && (w/h\*h)<16) print("severely underweight")

if ((w/h\*h)>=16 && (w/h\*h)<18.5) print("underweight")

if ((w/h\*h)>=18.5 && (w/h\*h)<25) print("Normal(healthy weight)")

if ((w/h\*h)>=25 && (w/h\*h)<30) print("Overweight")

if ((w/h\*h)>=30 && (w/h\*h)<35) print("Obese Class I(Moderately obese")

if ((w/h\*h)>=35 && (w/h\*h)<40) print("Obese Class II(Severely obese")

else if ((w/h\*h)>40) print("Obese Class III (Very severely obese)")

x<-(paste0("The weight can be in pound and the height in inches:", 703))

return(x)

}

1. **Write a function called sum\_of\_cubes, that calculates the sum of cubes of the first n natural numbers :**
2. if we have two numbers : 1, 2 then sum of squares is 9 ( 1^3 + 2^3)
3. if we have three numbers : 1, 2, 3 then sum of squares is 36 ( 1^3 + 2^3 + 3^3)

a) sum\_of\_cubes<- function(x,y) return(x^3+y^3)

b) sum\_of\_cubes<- function(x,y) return(x^3+y^3)

**5.) Write a function to calculate the mode (highest frequency) of the following vector:**

**x = c(2,3,3,4,4,5,6,7,9,10)**

X<- c(2,3,3,4,4,5,6,7,9,10)

table(X)

**6.) Write a function to calculate the no. of prime numbers of the following vector :**

**x = c(2,2,3,3,4,5,7,11,15,19,24,29)**

a<-c(2,2,3,3,4,5,7,11,15,19,24,29)

is.prime <- function(n) n == 2L || all(n %% 2L:max(2,floor(sqrt(n))) != 0)

count<- function(x)

{

x==0

for (a in x)

{if (is.prime2(a)) x=x+1

}

return(x)

}

**8.)Perform below operations using Data.frame and Data.table**

F1<- data.frame(empno=1:10,deptid=11:20, mgrid=(LETTERS[1:10]),stringsAsFactors = F)

require(data.table)

F2<- data.table(empno=1:5,11:15,sal=c(10000,12000,9000,32000,22000,8500,13000,15000,18000,20000),DOJ=c("jan2014","may2014","jan2014","jun2015","apr2016","dec2013","aug2106","sep2015","mar2015","jan2013"))

1. library(readr)

F1 <- read\_csv("C:/Users/Anjana/Desktop/F1.csv")

View(F1)

library(readr)

F1 <- read\_csv("C:/Users/Anjana/Desktop/F1.csv")

View(F2)

comp<-merge(F1,F2, by="empno")

comp1<-merge(F1,F2,all.x=TRUE)

comp2<-merge(F1,F2,all.y=TRUE)

comp3<-merge(F1,F2,all=TRUE)

subset(comp1,empno="NA")

library(dplyr)

New<- comp1 %>% group\_by(DOJ)%>%

select(DOJ,empno,deptid,mgr\_id,sal)%>%

summarise(totalsal=sum(sal),avgsal=mean(sal))

Hkl

empname<- c("raj","Mohan","somu","Ritu","Sita","Gulam","Ratan","Komal","KaranVeer","Hansraj")

attach(F2)

F2$empname<- empname

detach(F2)

**9.) Create R functions for the following operations**

1. library(dplyr)

Std1<- student %>% group\_by(classid)%>%

select count(distinct stdid)

1. library(dplyr)

Std1<- student %>% group\_by(classid,sectionid)%>%

select count(distinct stdid)

library(dplyr)

1. Std2<- student[duplicated(student$classid)

**10.) Create R functions for the following operations**

1. Find out if there are any nulls in a dataset or in some specific number of columns

Ans= is.na(Y)

1. Write a function to read data from hdfs and dump it back to hdfs ?

Ans=

Library(rhdfs)

HADOOP\_CMD=/Anjana/local/hadoop-2.6.0/bin/hadoop

hdfs.init()

hdfs.put

**11) a.) Remove duplicates from a given vector and return it back.**

dupt = c(1, 1, 5, 5, 2, 2, 6, 6, 1, 3)

dupt[duplicated(dupt)]

1. Compute count of distinct

unique(dupt)

1. Concatenate two strings.

one<- "My Life"

two<- "My Family"

nstring<- paste(one,two, sep="-")

1. Perform Column-wise/Row-wise sum using apply function.

a<- 3:10

cmat<- matrix(a,nrow=4)

apply(cmat,2,sum)

apply(cmat,1,sum)

1. Get list of files in an hdfs path.

Ans= hdfs.ls(‘/’)

1. Delete a file from hdfs if it exists.

Ans= hdfs.delete(“/RHadoop”)

**12.) Create R functions for the following operations**

a. Load some csv data from hdfs.

Ans- data <- read.table.ffdf(file=" C:/Users/Anjana/Desktop/F1.csv", nrows=50)

c.) Rename column names in a dataframe

name<- c("Annu","Guddu","Ranu", "somya")

age<-c(23,43,14,20)

d<- data.frame(name,age,stringsAsFactors = FALSE)

colnames(d)<- c("FName","Age in Number")

d. Drop given column from a data frame

d$`Age in Number`=NULL

e. Illustrate the difference between NA, NULL, NaN.

**NULL** represents the null object, it's a reserved word. NULL is perhaps returned by expressions and functions, so that values are undefined.

**NA** is a logical constant of length 1 which contains a missing value indicator. NA can be coerced to any other vector type except raw. There are also constants NA\_integer\_, NA\_real\_, NA\_complex\_ and NA\_character\_ of the other atomic vector types which support missing values: all of these are reserved words in the R language.

**NaN** means “not a number” is 0/0.

Therefore, NaN ≠ NA and there is a need for NaN and NA. is.na() returns TRUE for both NA and NaN, but is.nan() returns TRUE for NaN (0/0) and FALSE for NA.

**f. Return true if and only if all rows of given vector satisfy a certain condition.**

dat <- data.frame(x=c("a", "b", NA))

> sum(dat$x=="a", na.rm=T)

**g. Compute number of unique combinations in a data frame grouped by certain columns.**

class<-c("first","first","second","second","first","third","third")

marks<-c(45,55,20,67,60,80,75)

category<-data.frame(class,marks,stringsAsFactors = FALSE)

aggregate(.~class,category,sum)