JAMAL MOHAMED COLLEGE(AUTONOMOUS)

R-PROGRAMMING LAB (III - B.C.A -D)

1.Aim:

To Read two vectors V1 and V2 containing values (49,21,34,53,11) and (14,49,53,34,81). Now find out the values of V1 that are not present in V2 and store it into a new vector without using any predefined function.

Coding:

```
v1<-c(49,21,34,53,11)
v2<-c(14,49,53,34,81)
x<-v1[!v1 %in% v2]
cat("Difference of v1 is : ",x)
```

Output:

```
Difference of v1 is :  21 11>
```

2.Aim:

To Create a user defined function that performs the binary search on a numeric vector.

Coding:

```
binarySearch = function(vec,target) {
start <- 1; len <- length(vec)
while (start <= len){
    mid <- as.integer(round((start + len) / 2))
    if (abs(vec[mid] - target) ==0) {
        return(mid)
    } else if (vec[mid] < target) {
        start <- mid + 1
    } else {
        len <- mid - 1
    }
}
return(0)
```

```
 }


 vec <- c(4, 0, 3, 1, 5, 6, 2)
 sorted_vec <- sort(vec)
 target <- 4


 cat("Numeric vector is : ", vec, "\nSorted vector : ",
sorted_vec,"\ntarget = ", target, "\n")


 index <- binarySearch(sorted_vec, target)
 if (index!=0){
     cat("Element is present at index ", index, "\n")
 }else{
     cat("element not found")
 }
```

Output:

```
     Numeric vector is :  4 0 3 1 5 6 2
     Sorted vector :  0 1 2 3 4 5 6
     target =  4
     Element is present at index  5
```

3.Aim:

To create two 3 X 3 matrices A and B and perform the following operations a) Transpose of the matrix b) addition c) subtraction

Coding:

```
     A <- matrix(1:6, nrow=3, ncol=3)
     print(A)
     B <- matrix(7:12, nrow=3, ncol=3)
     print(B)


     #transpose of the matrix
     t1 <- t(A)
     print(t1)
```

```
t2 <- t(B)
print(t2)
#addition of two matrix
add <- A+B
print(add)
#subtraction of two matrix
sub <- A-B
print(sub)
```

Output:

```
     [,1] [,2] [,3]        #print(A)
[1,]    1    4    1
[2,]    2    5    2
[3,]    3    6    3
     [,1] [,2] [,3]        #print(B)
[1,]    7   10    7
[2,]    8   11    8
[3,]    9   12    9
     [,1] [,2] [,3]        #transpose of A
[1,]    1    2    3
[2,]    4    5    6
[3,]    1    2    3
     [,1] [,2] [,3]        #transpose of B
[1,]    7    8    9
[2,]   10   11   12
[3,]    7    8    9
     [,1] [,2] [,3]        #addition
[1,]    8   14    8
[2,]   10   16   10
[3,]   12   18   12
     [,1] [,2] [,3]        #subtraction
[1,]   -6   -6   -6
[2,]   -6   -6   -6
[3,]   -6   -6   -6
```

4.Aim:

To create a data frame that stores some basic information of laptop such as the configuration of laptop from at least five companies. Apply length(), str(), summary(), duplicated(), unique() functions or other functions on the data frame.

Coding:

```
laptops <- data.frame(
Company = c("Dell", "HP", "Lenovo", "Asus", "Dell"),
Processor = c("Intel Core i5", "AMD Ryzen 5", "Intel Core i7", "AMD Ryzen 7", "Intel Core i5"),
RAM = c(8, 16, 32, 16, 8),
Storage = c(256, 512, 128, 1024, 256)
)
print(laptops)
print(length(laptops))
print(str(laptops))
print(summary(laptops))
print(duplicated(laptops))
print(unique(laptops))
```

Output:

```
  Company   Processor RAM Storage              #dataframe of
1    Dell Intel Core i5   8     256                    laptops
2      HP   AMD Ryzen 5  16     512
3  Lenovo Intel Core i7  32     128
4    Asus   AMD Ryzen 7  16    1024
5    Dell Intel Core i5   8     256


[1] 4                                           #length


'data.frame':   5 obs. of  4 variables:
 $ Company  : chr  "Dell" "HP" "Lenovo" "Asus" ...
 $ Processor: chr  "Intel Core i5" "AMD Ryzen 5" "Intel Core i7" "AMD Ryzen 7" ...
 $ RAM      : num  8 16 32 16 8
```

```
 $ Storage  : num  256 512 128 1024 256
NULL                              #str(inner structure of dataframe)

   Company           Processor             RAM        Storage
 Length:5           Length:5          Min.   : 8   Min.   : 128.0
 Class :character   Class :character   1st Qu.: 8   1st Qu.: 256.0
 Mode  :character   Mode  :character   Median :16   Median : 256.0
                                       Mean   :16   Mean   : 435.2
                                       3rd Qu.:16   3rd Qu.: 512.0
                                       Max.   :32   Max.   :1024.0
                                             #summary
[1] FALSE FALSE FALSE FALSE  TRUE            #duplicated
  Company     Processor RAM Storage          #unique
1    Dell Intel Core i5   8     256
2      HP   AMD Ryzen 5  16     512
3  Lenovo Intel Core i7  32     128
4    Asus   AMD Ryzen 7  16    1024
```

5.Aim:

To create a data frame that stores the temperature of 10 cities along with their names. Using the function rownames(), put suitable names of the rows and columns of the data frame.

Coding:

```
cities <- c("Karur", "Trichy", "Namakkal", "Salem", "Coimbatore", "Madurai", "Villupuram", "Erode", "Tanjavore", "Ooty")
temperature <- c(70, 75, 80, 85, 90, 95, 100, 105, 110, 115)
x <- data.frame(Cities = cities,Temperature = temperature)

rownames(x) <- c("City-1","City-2","City-3","City-4","City-5","City-6","City-7","City-8","City-9","City-10")
print(x)

colnames(x) <- c("Cities","Temperature (F)")
print(x)
```

Output:

```
            Cities     Temperature           #rowname
City-1       Karur              70
City-2      Trichy              75
City-3    Namakkal              80
City-4       Salem              85
City-5  Coimbatore              90
City-6     Madurai              95
City-7  Villupuram             100
City-8       Erode             105
City-9   Tanjavore             110
City-10       Ooty             115

            Cities Temperature (F)           #colnames
City-1       Karur              70
City-2      Trichy              75
City-3    Namakkal              80
City-4       Salem              85
City-5  Coimbatore              90
City-6     Madurai              95
City-7  Villupuram             100
City-8       Erode             105
City-9   Tanjavore             110
City-10       Ooty             115
```

6.Aim:

To create a list that stores some arbitrary numbers as components. Add three new numbers on the list and delete the third and eighth number of the list.

Coding:

```
x <- list(1, 2, 3, 4, 5, 6, 7, 8, 9)
x <- c(x, 10, 11, 12)
print(x)
```

```
x <- x[-c(3, 8)]
print(x)
```

Output:

```
[[1]]    [[2]]    [[3]]    [[4]]    [[5]]    [[6]]
[1] 1    [1] 2    [1] 3    [1] 4    [1] 5    [1] 6
[[7]]    [[8]]    [[9]]    [[10]]   [[11]]   [[12]]
[1] 7    [1] 8    [1] 9    [1] 10   [1] 11   [1] 12

[[1]]    [[2]]    [[3]]    [[4]]    [[5]]    [[6]]
[1] 1    [1] 2    [1] 4    [1] 5    [1] 6    [1] 7
[[7]]    [[8]]    [[9]]    [[10]]
[1] 9    [1] 10   [1] 11   [1] 12
```

7.Aim:

To Check whether a year (integer) entered by the user is a leap year or not.

Coding:

```
year = as.integer(readline(prompt="Enter a year: "))


if((year %% 4) == 0){
   if((year %% 100) == 0){
      if((year %% 400) == 0){
         print(paste(year," is a leap year"))
      } else {
         print(paste(year," is not a leap year"))
      }
   } else {
      print(paste(year," is a leap year"))
   }
} else {
   print(paste(year," is not a leap year"))
}
```

Output:

```
Enter a year: 2000
2000 is a leap year
```

8.Aim:

To create a recursive function that generates the Fibonacci series.

Coding:

```
recurse_fibonacci <- function(n){
if(n <= 1){
return(n)
} else {
return(recurse_fibonacci(n-1) + recurse_fibonacci(n-2))
}
}
nterms = as.integer(readline(prompt="How many terms? "))
if(nterms <= 0){
print("Enter a positive integer")
} else {
print("Fibonacci Series are : ")
for(i in 0:(nterms-1)){
print(recurse_fibonacci(i))
 }
}
```

Output:

```
How many terms? 7
[1] "Fibonacci Series are : "
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
```

9.Aim:

To create two vectors where one vector contains positive values and the other contains negative values. Find the correlation between the two vectors.

Coding:

```
positives <- c(1, 2, 3, 4, 5)

negatives <- c(-1, -2, -3, -4, -5)

cor(positives, negatives)              #cor()-function for
                                    finding the co-relation
                                    between two vectors
```

Output:

```
[1] -1
```

10.Aim:

To extract first 10 English letters in lower case and last 10 letters in upper case and extract letters between 22nd to 24th letters in upper case.

Coding:

```
print("First 10 letters in lower case: ")

x = head(letters, 10)

print(x)

print("Last 10 letters in upper case: ")

y = tail(LETTERS, 10)

print(y)

print("letters between 22nd and 24th letters in upper case:")

z = tail(LETTERS[22:24])

print(z)
```

Output:

```
[1] "First 10 letters in lower case"

[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"

[1] "Last 10 letters in upper case"

[1] "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"

[1] "letters between 22nd and 24th letters in upper case:"

[1] "V" "W" "X"
```

11.Aim:

To Get the unique elements of a given string and unique numbers of vector.

Coding:

```
string <- "abcabcdefg"
unique(string)


vec <- c(1, 2, 2, 3, 3, 3, 4, 4, 4, 4)
unique(vec)
```

Output:

```
[1] "a" "b" "c" "d" "e" "f" "g"
[1] 1 2 3 4
```

12.Aim:

To Illustrate the use of regular expressions.

Coding:

```
# Extract all the digits from a string
string <- "The quick brown fox jumps over the lazy dog 1234567890"
matches <- regexpr("[0-9]", string)
digits <- regmatches(string, matches)


# Extract all the words that start with "T" from a string
string <- "The quick brown fox jumps over the lazy dog"
matches <- regexpr("T\\w+", string)
words <- regmatches(string, matches)


# Replace all the occurrences of "the" with "THE" in a string
string <- "The quick brown fox jumps over the lazy dog"
string <- gsub("the", "THE", string)
```

Output:

```
[1] 1234567890
[1] "The" "the"
[1] "The quick brown fox jumps over The lazy dog"
```

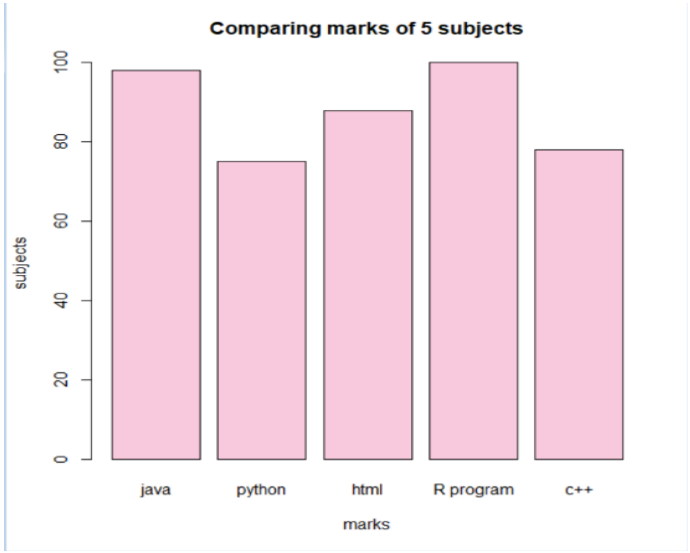
13.Aim:

To Create a simple bar plot of five subjects' marks.

Coding:

```
marks <- c(80, 75, 90, 70, 85)
barplot(marks,
main = "Comparing marks of 5 subjects",
xlab = "marks",
ylab = "subjects",
names.arg = c("java","python","html","R program","c++"),
col = "#F8C8DC",
horiz = FALSE)
```

Output:



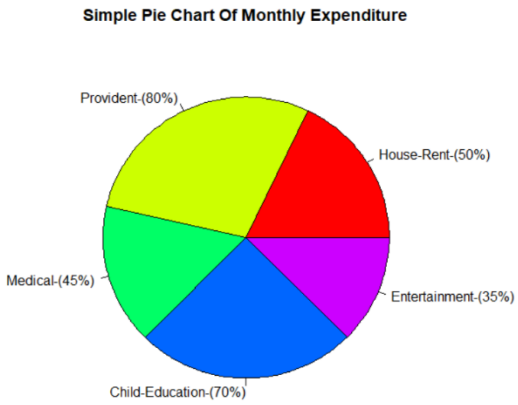
14.Aim:

To create a simple pie chart of monthly expenditure.

Coding:

```
expenses <- c(500,800,450,700,350)
reason <- c("House-Rent-(50%)","Provident-(80%)","Medical-
(45%)","Child-Education-(70%)","Entertainment-(35%)")
pie(expenses,reason,main = "Simple Pie Chart Of Monthly
Expenditure",col = rainbow(length(expenses)))
```

Output:



15.Aim:

To Draw an empty plot and an empty plot specify the axes limits of the graphic.

Coding:

```
x <- rnorm(100)
y <- rnorm(100)
plot(x, y)
xlim <- c(-2, 2)
ylim <- c(-5, 5)
```

Output: