Experiment-4

a) Various operations on Vectors

Vectors contain a sequence of homogenous types of data. if mixed values are given then it auto converts the data according to precedence

A creating a vector

```
> x<-c(1,5,4,9,0)
> type of(x)
  [1] "double"
> length(x)
  [1] 5
> x<-c(1,5.4,TRUE,"hello")
> x
[1] "1" "5.4" "TRUE" "hello"
> type of(x)
  [1] "character"
```

creating a vector using : operator

```
> x<-1:7
> x
[1] 1 2 3 4 5 6 7
> y<-2:-2
> y
[1] 2 1 0 -1 -2
```

Creating a vector using seq() function

```
>seq(1,3,by=0.2) # Specify step size.
 [1] 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6
     2.8 3.0
```

b. Acess elements of a Vector:-

using integer vector as index

```
> x
[1] 0  2  4  6  8  10
> x[3] # access 3rd element
[1] 4
> x[c(2,4)] # access 2nd & 4th element
[1] 2 6.
> x[1] # acess all the but 1st element
> x[c(2,-4)] # cannot mix +ve & -ve integers.
> x[c(2,4,3,5 4)] # real numbers are
                  truncated to integs.
```

using logical vector as index:-

logical vector for indexing, the position where the logical vector is TRUE.

```
>x[c(TRUE, FALSE, FALSE, TRUE)]
 [1] -3 3
>x[x<0] # filtering vectors based on condition
> x[x>0]
```

Using character Vectors as Index:-

This type of indexing is useful dealing with named Vectors.

```
> x <- c ("first" = 3, "second" = 0, "third" = 2)
> names(x)
> x["second"]
> x[c("first", "third")]
```

How to modify Vector?

We can Modify a Vector using the assigned operator.

```
> x
> x[2] <- 0;
> x          # modify second element
> x[x<0] <- 5;
> x          # modify elements less than 0
> x <- x[1:4]
> x # truncate x to first 4 elements.
```

Deleting a Vector.

```
# creating a vector
  x <- c(5,2,1,6)
# deleting a vector
  x <- NULL
print("deleted Vector")
print(x)
```

Arithematic operations:—

```
# creating Vectors.
  x <- c(5,2,5,1,5,1,2)
```

```
y <- c(7,9,1,5,2,1)
# addition
  z <- x+y
> print ("addition")
> print (z)
# Subtraction
  z <- x-y
> print("Subtraction")
> print (z)
```

Soting of vectors :-

```
#creating a vector
x <- c(5,2,5,1,51,2)
# sort in ascending order
A <- sort(x)
print ("sorting done in ascending order")
print(A)
#sort in desending order
B <- sort (x, decreasing = TRUE)
print ("sorting done in descending order")
Print (B).
```

4(b) Finding the sum and average of given numbers using arrays.

We can use the array() function to create an array, and the dim parameter to specify the dimensions.

An array with one dimension with values ranging from 1 to 24.

```
> thisarray <- (1:24)
> thisarray
```

array with more than one dimension

```
multiarray <- array(thisarray, dim = c(4,3,2))
multiarray.
```

Access Array Items

```
Multiarray[2,3,2]
[1]22
```

Access all the items from the first row from matrix one

```
Multiarray <- array(thisarray, dim = c(4,3,2))
multiarray[c(1),,1]
```

Access all the items from the first column from matrix one

```
Multiarray <- array(thisarray, dim(4,3,2))
multiarray[,c(1),1]
```

Chek if an item exists: To find out if a specified item is present in an array, use %in% operator

2 %in% multiarray

use the dim() function to find out the amount of rows and columns in an array

>dim (multiarray)

use the length() function to find the dimension of an array

>length (multiarray)

Loop through array:-

You can loop through the array items by using a for loop

```
thisarray <- c(1:24)
multiarray <- array (thisarray, dim = c(4,3,2))
for (x in multiarray) {
  print(x)
}
```

Program 6)

Finding the sum and average of given numbers using arrays

```
this array <- C(1:24)
multiarray <- array (this array ,dim = c(4,3,2))
 Sum <- 0
 Count <- 0
 avg <- 0.0
for(x in multiarray){
  print(x)
 Count = count + 1
  Sum = Sum + x
}
Cat ("Sum of the numbers in the array:",
                        Sum)
 avg = Sum / count
Cat ("\n Average of the numbers in the
          array:", avg)
```

4(c) To display elements of list in reverse order

```
#create list and store the elements
mylist <- list ("apple ","banana","cherry","kiwi",
    "orange","melon","mango")
# To find out how many items a list has,
  use the length() function
  x<-length(mylist)
# Display the elements in the reverse order
  print(Mylist[x:1])
```

4(d)

Finding the minimum and maximum elements in the array.

```
# R program to illustrate the use of max() func
# creating a matrix
arr = array (2:13, dim = c(2,3,2))
print (arr)

# using max() function
mx <- min (arr)
cat ("maximum element in the array", mx)
# using min() function
mn <- min (arr)
cat ("In minimum element in the array",
      mn)
> source ("/cloud/project/max_min_array.r")
```