Exp. :4 Demonstrate Array operations in R

- **Aim:** To create Arrays, access and manipulate the elements of Array.

- **Arrays:**
  - An Array is a **multidimensional vector.**
  - It must all be of the same type and individual elements are **accessed** in a similar fashion **using square brackets.**
  - An array in R can have **one, two or more dimensions.**
  - A two-dimensional array is the similar to a matrix.
  - Array can have **multiple dimensions while matrix has only two dimensions.**

Exp. :4 Demonstrate Array operations in R

- **Aim:** To create Arrays, access and manipulate the elements of Array.
- **Creating Arrays:**
- An array is created using the **array()** function. It takes vectors as input and uses the values in the **dim** parameter to create an array.

  - myarray <- **array(data, dim, dimnames)**

  - **data** is the input vector which becomes the data elements of the matrix.

  - **dim** is a vector of dimensions. **Ex:** c(2,3,4) then it creates 4 rectangular matrices each with 2 rows and 3 columns. Arrays can store only data type.

Exp. :4 Demonstrate Array operations in R

- **Aim:** To create Arrays, access and manipulate the elements of Array.

- **Creating Arrays:**
  - v <- array(1:12, c(2, 3, 2))

  - **# it produces the following result :**
  - **, , 1**
  - [,1] [,2] [,3]
- [1,] 1 3 5
- [2,] 2 4 6

- **, , 2**
- [,1] [,2] [,3]

Exp. :4 Demonstrate Array operations in R

- **Aim:** To create Arrays, access and manipulate the elements of Array.

```
# Create two vectors of different lengths.
      vector1 <- c(10,20,30,40,50,60)
      vector2 <- c(7,8,9)
# Take these vectors as input to the array.
      result <- array(c(vector1,vector2),dim = c(3,3,2))
      print(result)
#it produces the following result −
, , 1
     [,1] [,2] [,3]
[1,]   10   40    7
[2,]   20   50    8
[3,]   30   60    9
, , 2
     [,1] [,2] [,3]
[1,]   10   40    7
[2,]   20   50    8
```

Exp. :4 Demonstrate Array operations in R

- **Aim:** To create Arrays, access and manipulate the elements of Array.

  - Using the **dimnames** parameter, we can give names to the rows, columns and matrices in the array .

```
# Naming Rows and Columns
    v1 <- c(10,20,30,40,50,60)
    v2 <- c(7,8,9)
    rownames <- c("Row1","Row2","Row3")
    colnames <- c("Col1","Col2","Col3")
    matrixnames <- c("Matrix1","Matrix2")
# Take these vectors as input to the array
result <- array(c(v1,v2),dim = c(3,3,2),dimnames=  list(rownames,colnames, matrixnames))
print(result)
```

Exp. :4 Demonstrate Array operations in R

- **Aim:** To create Arrays, access and manipulate the elements of Array.

- **# it will produces the following result –**

- , , Matrix1
- Col1 Col2 Col3
- Row1 10 40 7
- Row2 20 50 8
- Row3 30 60 9

- , , Matrix2
- Col1 Col2 Col3
- Row1 10 40 7
- Row2 20 50 8

Exp. :4 Demonstrate Array operations in R

- **Accessing Array Elements :**
- Use the indexes (starting with 1) to access a row or a column or an element.

- **Example 1:** A <- array(1:12, dim=c(2,2,3))

- **A[1, ,]** **# first element(row) of all columns of all matrices**

- **#it produces the following result −**

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11
```

Exp. :4 Demonstrate Array operations in R

- A[,2 ,] # All rows second element(column) of all matrices
- #it produces the following result −

	[,1]	[,2]	[,3]
[1,]	3	7	11
[2,]	4	8	12

- A[, ,3] # All rows and All columns of third matrix
- #it produces the following result −

	[,1]	[,2]
[1,]	9	11
[2,]	10	12

Exp. :4 Demonstrate Array operations in R

Example 2 on Accessing Array Elements:

```
> v1 <- c(10,20,30,40,50,60)
> v2 <- c(7,8,9)
> rownames <- c("Row1","Row2","Row3")
> colnames <- c("Col1","Col2","Col3")
> matrixnames <- c("Matrix1","Matrix2")
> result <- array(c(v1,v2),dim = c(3,3,2),dimnames=   list(rownames,colnames, matrixnames))
> print(result)
```

 # Print the third row of the second matrix of the array.

```
print(result[3,,2])
```

 # It will produce the following result.

Col1	Col2	Col3
30	60	9

Exp. :4 Demonstrate Array operations in R

Example 2 on Accessing Array Elements:

```
# Print the 2nd Matrix.
print(result[,,2])

# It will produce the following result.
```

	Col1	Col2	Col3
Row1	10	40	7
Row2	20	50	8
Row3	30	60	9

```
# Print the element in the 1st row and 3rd column of the 1st matrix.
print(result[1,3,1])

# It will produce the following result.
7
```

Exp. :4 Demonstrate Array operations in R

Example 3 on Accessing subset of array elements

A smaller subset of the array elements can be accessed by defining a range of row or column limits.

```
a = array(1:15, dim = c(2, 4, 2))

# print elements of both the rows and columns 2 and 3 of matrix 1
print (a[, c(2, 3), 1])
```

 # It will produce the following result.

```
     [,1] [,2]
[1,]    3    5
[2,]    4    6
```

Exp. :4 Demonstrate Array operations in R

- **Array Addition and Subtraction:**

```
v1 <- c(10,20,30,40,50,60)
v2 <- c(7,8,9)
result <- array(c(v1,v2),dim = c(3,3,2))
#To perform the arithmetic operations, we are converting the multidimensional matrix into a one-dimensional matrix
A1=result[,,1]
A2=result[,,2]

print(A1+A2)                    # Addition
print(A1-A2)                    # Subtraction
```

Exp. :4 Demonstrate Array operations in R

Calculations Across Array Elements:

- We can do calculations across the elements in an array using the **apply()** function.

Syntax: apply(x, margin, fun)

- **x** is an array.
- **MARGIN:** It takes a value or range between 1 and 2 to define where to apply the function:
  - **-MARGIN=1:** the manipulation is performed on rows
  - **-MARGIN=2:** the manipulation is performed on columns.
- **fun:** Tells which function to apply. Built-in functions **like mean, median, sum, min, max** and even user-defined functions can be applied, fun is the function to be applied across the elements of the array.

Exp. :4 Demonstrate Array operations in R

- **Calculations Across Array Elements:**

Example:

```
A=array(1:6,dim=c(2,3))
apply(A, c(2), sum)
```

 # It will produce the following result.

```
[1] 3       7       11
```

Exp. :4 Demonstrate Array operations in R

- **aperm() function – Array Transposition:**
  Permute the dimensions of an array.

#Permute the dimensions of an array.

```
A=array(1:6,dim=c(2,3))
P=aperm(A)
apply(P, c(2), sum)
```

 # It will produce the following result.

```
[1] 9        12
```

Adding elements to array:

Function Name	Description
c(vector, values)	c() function allows us to append values to the end of the array. Multiple values can also be added together.
append(vector, values)	This method allows the values to be appended at any position in the vector. By default, this function adds the element at end.
append(vector, values, after=length(vector))	adds new values after specified length of the array specified in the last argument of the function.
length function	Elements can be added at length+x indices where x>0.

Examples on Adding elements to array:

 # creating an array

```
x <- c(10, 20, 30, 40, 50)
```

- **# addition of element using c() function**

```
x <- c(x, 1)
print ("Array after 1st modification ")
print (x)
```

- **# It will produce the following result.**

- **# addition of element using append() function**

```
x <- append(x, 2)
print ("Array after 2nd modification ")
print (x)
```

- **# It will produce the following result.**
- [1] 10 20 30 40 50 1 2

Examples on Adding elements to array:

 # creating an array

```
x <- c(10, 20, 30, 40, 50)
```

- **# adds new elements after 3rd index**
- `print ("Array after 3rd modification ")`
- `x <- append(x, c(-6, -6), after = 3)`
- `print (x)`

- **# It will produce the following result.**
- `[1] 10 20 30 -6 -6 40 50 1 2`