In R, Missing values are represented by the symbol NA (not applicable). Impossible values (domain errors like division by 0 et logs of negative numbers are represented by the symbol NAN (not-A-Number).

5MARKS ANSWERS:

(1) DATA TYPES:

(a) CHARACTER:

R supports character data types where you have all the alphabets and special characters. It stores character values or strings. Strings in R can contain alphabets, numbers, and symbols.

PROGRAM:

```
char = "Sonali"
print(class(char)) # print the class of name of char
print(type of (char))
```

OUTPUT:

```
[1] "character"
[1] "character"
```

(b) COMPLEX:

R supports complex data types which are set of all the complex numbers. The complex data type is to store numbers with an imaginary component.

PROGRAM:

```
x= 4+3i
Print (class(x))
Print (type of (x))
```

OUTPUT:

[1] "complex"
[1] "complex"

(c) LOGICAL DATA TYPE:

R has logical data types that take either a value of true or false. A logical value is often created via a comparison between variables.

PROGRAM:

```
x=4
y=3
z= x>y
```

print(z)
print (class(z))
print (type of ( ))

OUTPUT:

[1] TRUE
[1] "logical"
[1] "logical"

(d) INTEGER:

R supports integer data types which are the set of all integers. You can create as well as convert a value into an integer type using as.integer( ) function. You can also use the capital 'L' notation as a suffix to denote that a numeric value is of the integer data type.

PROGRAM:

```
x=as.integer(5)
Print (class(x))
Print (type of (x))
y=5L
print(class(y))
print(type of (y))
```

OUTPUT:

[1] "integer"

[1] "integer"

[1] "integer"

[1] "integer"

(e) NUMERIC DATATYPE:

Decimal values are called numerics in R. It is the default data type for numbers in R. If you assign a decimal value to a variable x as follows, x will be of numeric type:

PROGRAM:

```
x=5.6
print(class(x))
print(typeof(x))
```

OUTPUT:

[1] "numeric"

[1] "double"

(2) DATA FRAME:

CHARACTERISTICS OF R DATA FRAME:

(1) The column names should be non-empty.

(2) The row names should be unique.

(3) The data frame can hold the data which can be an numeric, character or factor type.

(4) Each column should contain the same number of data items.

EXAMPLE:

```
employee_data ← data.frame(
  Employee_id = c(1:5),
  employee_name = c("James", "Harry", "Sherie", "Jim", "David"),
  Sal = c(642.3, 535.2, 681.0, 739.0, 925.26),
  Join_data = as.Date(c("2013-02-04", "2017-06-21",
                        "2012-11-14", "2018-05-19", "2016-03-25")),
  StringAsFactors = FALSE)
Print(employee_data)
Str(employee_data) → [: structure of R data frame]
```

(3) EXTRACTING FIRST TWO COLUMNS:

```
output<-employee_data[1:2,]
print(output)
```

(4) MATRIX MULTIPLICATION:

(1) matrix multiplication in R - element by element

```
X = matrix(c(1,3,5,7), ncol=2, nrow=2)
Y= matrix(c(2,4,6,8), ncol=2, nrow=2)
Z= matrix(c(5,7,3,1), ncol=2, nrow=2)
d<-x*y*z
print(x)
print(y)
print(z)
print(d)
```

output:

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

	[,1]	[,2]
[1,]	2	6
[2,]	4	8

	[,1]	[,2]
[1,]	5	3
[2,]	7	1

	[,1]	[,2]
[1,]	10	90
[2,]	84	56

(2.) matrix multiplication - %*% matrix multiplication between the two matrices

```
x= matrix(c(1,3,5,7), ncol=2, nrow=2)
y= matrix(c(2,4,6,8), ncol=2, nrow=2)
z= matrix(c(5,7,1,1), ncol=2, nrow=2)
d <- x %*% y %*% z
Print(x)
```

```
     [,1] [,2]
[1,]    2    6
[2,]    4    8

     [,1] [,2]
[1,]    5    3
[2,]    7    1

     [,1] [,2]
[1,]   10   90
[2,]   84   56
```

(2.) matrix multiplication - %*% matrix multiplication between the two matrices

```
x = matrix(c(1,3,5,7), ncol=2, nrow=2)
y = matrix(c(2,4,6,8), ncol=2, nrow=2)
z = matrix(c(5,7,3,1), ncol=2, nrow=2)
d <- x %*% y %*% z
print(x)
print(y)
print(z)
```

print(d)

OUTPUT:

	[,1]	[,2]
[1,]	1	5
[2,]	3	7

	[,1]	[,2]
[1,]	2	6
[2,]	4	8

	[,1]	[,2]
[1,]	5	9
[2,]	7	1

	[,1]	[,2]
[1,]	432	112
[2,]	688	176