1] Write a R Program for different types of data structures in R.:

```r
vec <- c(10, 20, 30, 40, 50)
print("Vector")
print(vec)


mat <- matrix(1:9, nrow = 3, ncol = 3)
print("matrix")
print(mat)


arr <- array(1:12, dim = c(3, 2, 2))
print("Array")
print(arr)

lst <- list(Name = "Basavaraj", Age = 21, Marks = c(90, 85, 8[illegible]))
print("List")
print(lst)


df <- data.frame(ID = c(1, 2, 3), Name = c("Ravi",
        "Sneha", "Kiran"), score = c(85, 92, 78))
print("Data Frame")
print(df)

colors <- factor(c("Red", "Blue", "Green", "Red", "Blue"))
print("Factor")
print(colors)
```

1] Output :

* [1] "Vector"

  [1] 10 20 30 40 50

* [1] "Matrix"

	[,1]	[,2]	[,3]
[1,]	1	4	7
[2,]	2	5	8
[3,]	3	6	9

* [1] "Array"

,,1

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

,,2

	[1]	[2]
[1]	7	10
[2]	8	11
[3]	9	12

* "List"

$ Name

[1] "Basavaraj"

$ Age

[1] 21

$ Marks

[1] 90 85 88

Scanned with CamScanner

1] Output:

* [1] "Data Frame"

	ID	Name	Score
1	1	Ravi	85
2	2	Sneha	92
3	3	Kiran	78

* [1] "Factor"

[1] Red Blue Green Red Blue
levels: Blue Green Red.

NBata

2] Write a program that include variables, constants, datatypes :

```
name <- "xyz"
age <- 22
height <- 5.4
is_student <- TRUE

print ("Variables")
print (name)
print (age)
print (height)
print (is_student)

PI <- 3.14159
GRAVITY <- 9.8

print ("Constants")
print (PI)
print (GRAVITY)

num_var <- 25
int_var <- as.integer (25)
char_var <- "Hello R"
log_var <- FALSE
complex_var <- 2 + 3i

print ("Data Types"
print (num_var)
print (int_var)
print (char_var)
print (log_var)
print (complex_var)
```

2]

```
print ("Checking Types")
print(class(num-var))
print(class(int-var))
print(class(char-var))
print(class(log-var))
print(class(complex-var))
```

2] <u>Output</u>:

```
[1] "Variables"
[1] "xyz"
[1] 21
[1] 5.4
[1] TRUE

[1] "Constants"
[1] 3.14159
[1] 9.8

[1] "Data Types"
[1] 25
[1] 25
[1] "Hello R"
[1] FALSE
[1] 2+3i

[1] "Checking Types"
[1] "numeric"
[1] "integer"
[1] "character"
[1] "logical"
[1] "complex"
```

NBate

3] Write R program that include different operators, control ~~statements~~ structures, default values for arguments, returning complex objects.

```
print ("Implementation of different operators, control
       structures, default values for arguments, returning
       complex objects")
add<- function (a,b = c(1, 3, 5, 7))
{
     res <- a+b
     return (res)
}
myvec1 <- c(2,4,6,8)
myvec2 <- c(11,12,13,14)
z <- add(myvec1,myvec2)
cat ("Result of vectors")
print (z)
result <- ifelse(test=(z %% 2 ==0), yes="Even", no="Odd"
print (result)
```

3] Output :

Implementation of different operators, control structures, default values, for arguements returning complex objects.

result of Vectors

[1] 13 16 19 22

[1] "odd" "Even" "Odd" "Even"

NBate

7] Write a R program that include linear algebra operations on vectors & Matrices.:

```
vec1 <- c(2,4,6)
vec2 <- c(1,3,5)

add_vec <- vec1+vec2
print("Vector Addition")
print(add-vec)

sub-vec <- vec1 - vec2
print("Vector Substraction")
print(sub-vec)

dot-product <- sum(vec1 * vec2)
print("Dot product of vec1 & vec2")
print(dot-product)

scalar-mul <- 3 * vec1
print("Scalar Multiplication (3*vec1):")
print(scalar-mul)

mat1 <- matrix(c(1,2,3,4), nrow=2, byrow=TRUE)
mat2 <- matrix(c(5,6,7,8), nrow=2, byrow=TRUE)

print("Matrix 1:")
print(mat1)
print("Matrix 2:")
print(mat2)
```

```
mat-add <- mat1 + mat2
print("Matrix Addition")
print(mat-add)

mat-sub <- mat1- mat2
print("Matrix Substraction")
print(mat-sub)

mat-elem-mul <- mat1 * mat2
print("Element-wise Multiplication")
print(mat-elem-mul)

mat-mul <- mat1 %*% mat2
print("Matrix Multiplication")
print(mat-mul)

mat-transpose <- det(mat1)
print("Determinant of Matrix1")
print(mat-transpose)

solve(mat1)
```

7] Output :

[1] "Vector Addition"

6 8 10 12

[1] "Vector Substraction"

-4 -4 -4 -4

[1] "Dot Product of vec1 & vec2"

70

[1] "Scalar Multiplication (3 * vec1)"

3 6 9 12

[1] "Matrix 1 :"

	[1]	[2]
[1]	1	2
[2]	3	4

[1] "Matrix 2 :"

	[1]	[2]
[1]	5	6
[2]	7	8

[1] "Matrix Addition :"

	[1]	[2]
[1]	6	8
[2]	10	12

7] Output :

```
[1] "Matrix Substraction"
      [1]   [2]
[1]   -4    -4
[2]   -4    -4

[1] "Element-Wise Multiplication :"
      [1]   [2]
[1]    5    12
[2]   21    32

[1] "Matrix Multiplication"
      [1]   [2]
[1]   19    22
[2]   43    50

[1] "Determinant of Matrix1 :"
   -2

> solve(mat1)
      [1]   [2]
[1]  -2.0   1.0
[2]   1.0  -0.5
```

NBate

8] Write a R program for any visual representation of an object with creating graphs using graphic functions: Plot(), Hist(), Line chart(), Pie(), Boxplot(), Scatterplots().

#Plot

```
print("Creating graph using plot() function()")
x<- c(10, 20, 30, 40, 50)
y<- c(35, 45, 55, 65, 75)
plot(x, y, col= "purple", xlab = "Heat", ylab = "Temp", pch=8,
     cex= 0.8, main="Relation Between Heat & Temprature")
```

#Histograph

```
print("Creating graph using Hist() function")
z<-c(2, 8, 5, 3, 0, 7, 6, 9, 1)
hist(z, col= "skyblue", xlab = "Distance", xlim = c(0,10),
ylim= c(0,5), breaks = 5, border = "red", main= "Histogram
 of Distance")
```

Line Chart

```
print("Line Chart")
p<- c(1, 2, 3, 4, 5, 6, 7)
q<- c(3, 8, 7, 5, 7, 1, 2)
plot(p, q, col= "blue", xlab= "Time", ylab= "Distance", main=
"Relation between Time and Distance", type="l")
```

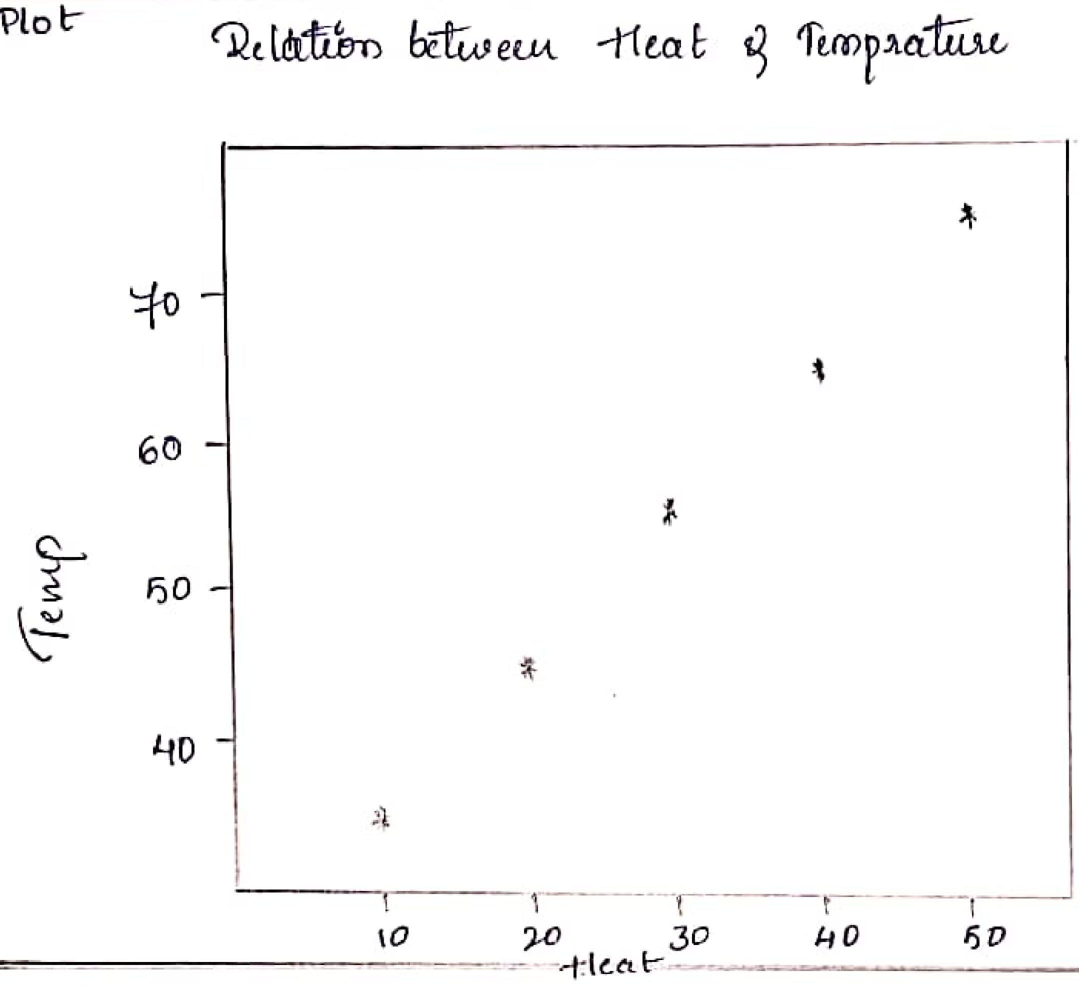
```
#Pie Chart
print("Pie Chart")
v1 <- c(5, 8, 9, 3, 4, 2)
v2 <- c("Graphes", "Mango", "Apple", "Kiwi", "Pineapple",
        "Watermelon")
pie(v1, labels = v2, radius = 1, clockwise = FALSE, init.angle =
      90, col = rainbow(length(v1)), main = "Fruits")
legend("bottomleft", v2, fill = rainbow(length(v1)), cex=0.6)

# Boxplot
print("Boxplot Graph")
x <- c(20, 38, 42, 25, 57, 48, 33, 54, 40, 28)
boxplot(x, col = "pink", ylab = "Weight of students", main =
        "Plotting Boxplot")
```

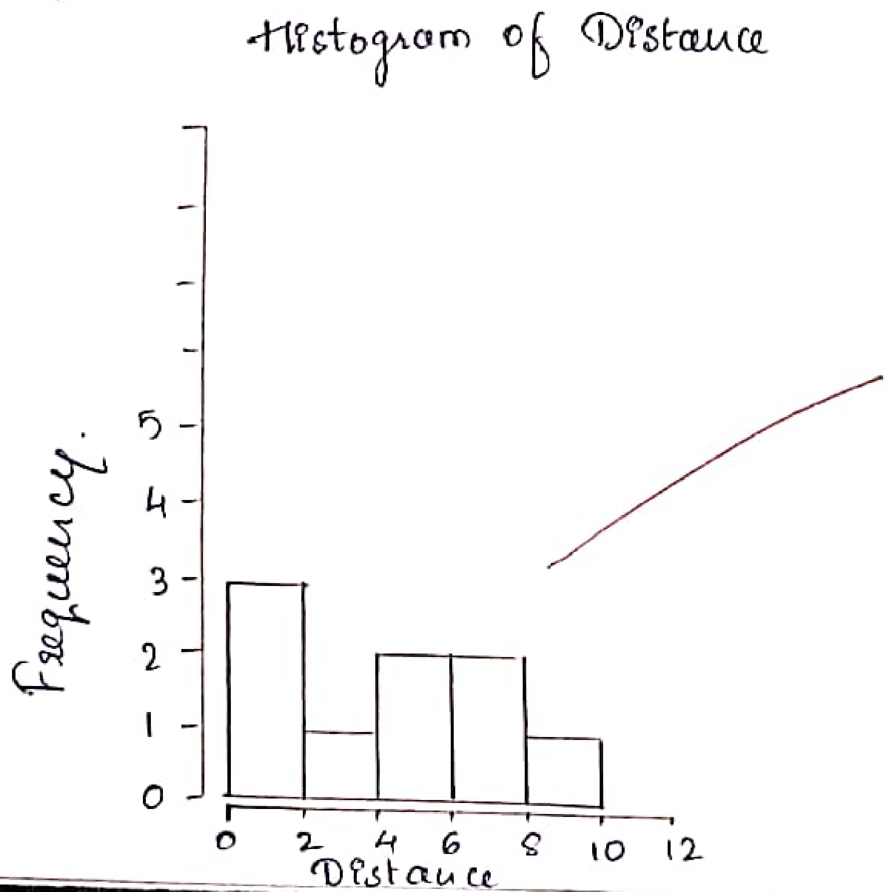
Scanned with CamScanner

8] Output:

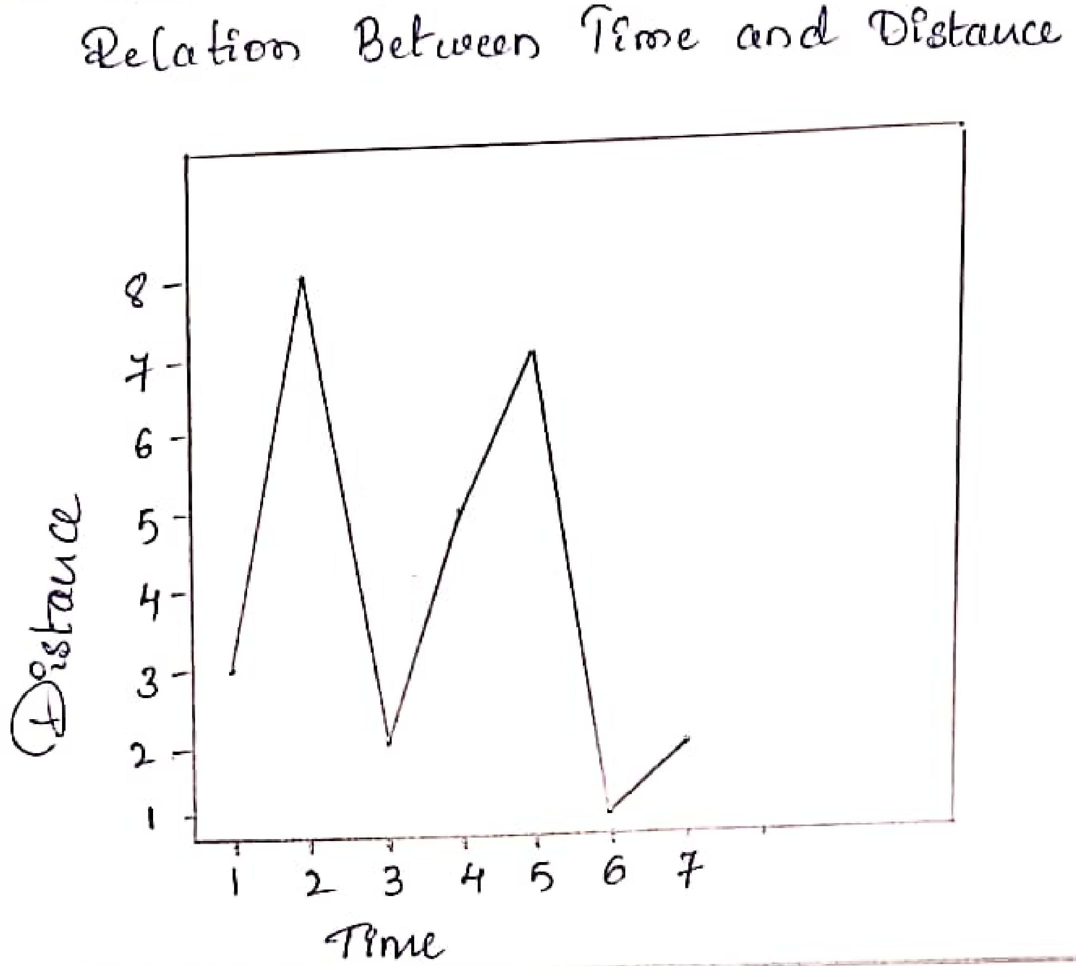
→ Plot



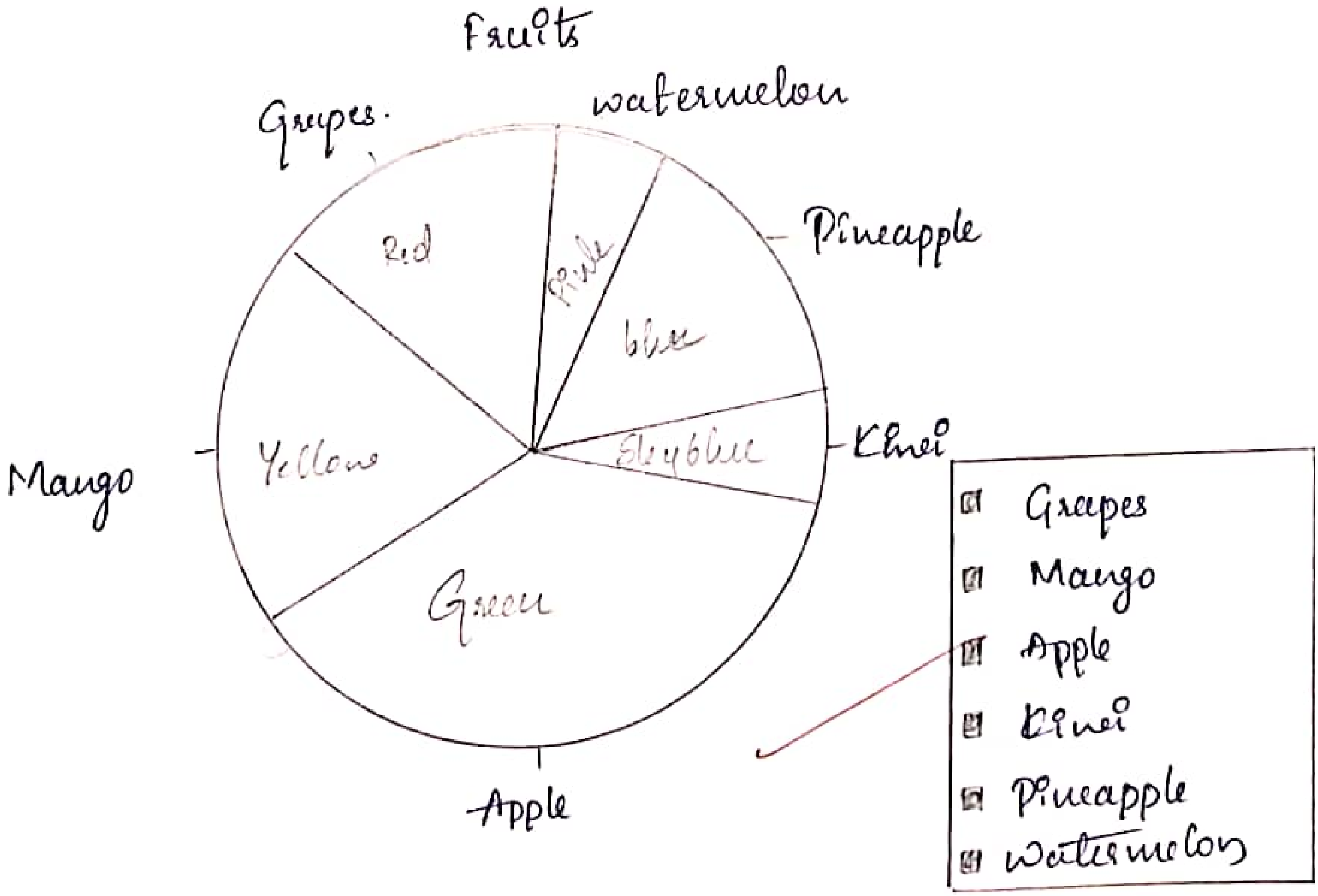
→ Histograph



→ Line Chart



→ Pie Chart



→ Boxplot