**Name – Shubham Dubey**
**PRN – 22070521139 , Sec – B**

**CODE:-**

**a) Create vector x and add 2 to it**
```
x <- 10:20
y <- x + 2
```

**b) Multiply y by 3**
```
z <- y * 3
```

**c) Subtract 6 from z and divide the result by 3**
```
answer <- (z - 6) / 3
```

**d) Print the answer variable**
```
print(answer)
```

**e) Do entire operation in a single line**
```
answer_single_line <- (((10:20) + 2) * 3 - 6) / 3
print(answer_single_line)
```

**f) What do you need to do to get the same result?**
**--> Use parentheses to respect operator precedence and replicate step-by-step computation**

**g) Do you notice anything about the operations?**
**Yes, operations follow BODMAS Rule mathematical operator precedence:**

**h) Create the following vectors using seq()**

**(i) 1, 1.5, 2, 2.5, …, 12**
```
v1 <- seq(1, 12, by = 0.5)
print(v1)
```

**(ii) 1, 8, 27, 64, …, 1000 (Cubes of 1:10)**
```
v2 <- (1:10)^3
print(v2)
```

**i) Write an R program to count a specific value in a given vector**
```
sample_vector <- c(2, 3, 5, 2, 6, 2, 8, 2)
specific_value <- 2
count <- sum(sample_vector == specific_value)
print(paste("Count of", specific_value, "is", count))
```

**j) Find common elements from multiple vectors**
```
vec1 <- c(1, 2, 3, 4, 5)
vec2 <- c(3, 4, 5, 6, 7)
vec3 <- c(0, 2, 3, 5, 9)
common_elements <- Reduce(intersect, list(vec1, vec2, vec3))
print("Common elements:")
print(common_elements)
```

**k) Create a symmetric vector (1 to 20, then back to 1)**

```
symmetric_vector <- c(1:20, 19:1)
print(symmetric_vector)
```

**m) Use grepl() to check which sentences contain the word "data"**

```
quotes <- c("Data is the new oil",
          "Big data means big responsibility",
          "Clean data is gold")

contains_data <- grepl("data", quotes, ignore.case = TRUE)
print(contains_data)
```

**OUTPUT:-**

```
[1] 10 11 12 13 14 15 16 17 18 19 20
 [1] 10 11 12 13 14 15 16 17 18 19 20
 [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0
[16]  8.5  9.0  9.5 10.0 10.5 11.0 11.5 12.0
 [1]    1    8   27   64  125  216  343  512  729 1000
[1] "Count of 2 is 4"
[1] "Common elements:"
[1] 3 5
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 19 18 17 16 15
[26] 14 13 12 11 10  9  8  7  6  5  4  3  2  1
[1] TRUE TRUE TRUE
```

## Practical – 2

**Name – Shubham Dubey**
**PRN – 22070521139 , Sec – B**

**CODE:-**

**a) Create a logical vector**

```
monster <- c(TRUE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE)
```

**b) Create a character vector yugioh and check type**

```
yugioh <- c("Blue-Eyes", "Dark Magician", "Red-Eyes", "Exodia")
print(typeof(yugioh))
```

**c) Combine monster and yugioh, check type**

```
combined <- c(monster, yugioh)
print(combined)
print(typeof(combined))
```

**d) Combine numeric and logical into a new vector**

```
atk <- c(3000, 2500, 2400, 1000)
coerce.check <- c(atk, monster)
```

```
print(coerce.check)
print(typeof(coerce.check))
```

e) Explicit Type Conversion using as.*

```
as_char <- as.character(monster)
as_num <- as.numeric(monster)
print(as_char)
print(as_num)


# Label Encoding

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
labels = ['Fire', 'Water', 'Fire', 'Electric']
encoded = le.fit_transform(labels)
print(encoded)

# One-Hot Encoding
from sklearn.preprocessing import OneHotEncoder
import numpy as np
ohe = OneHotEncoder(sparse=False)
one_hot = ohe.fit_transform(np.array(labels).reshape(-1,1))
print(one_hot)
```

Output :-

```
[1] "character"
 [1] "TRUE"         "TRUE"         "TRUE"         "FALSE"
 [5] "TRUE"         "TRUE"         "TRUE"         "TRUE"
 [9] "TRUE"         "TRUE"         "Blue-Eyes"     "Dark Magician"
[13] "Red-Eyes"      "Exodia"
[1] "character"
 [1] 3000 2500 2400 1000    1    1    1    0    1    1    1    1    1    1
[1] "double"
 [1] "TRUE"  "TRUE"  "TRUE"  "FALSE" "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE" ERROR!

[10] "TRUE"
 [1] 1 1 1 0 1 1 1 1 1 1
Error: unexpected symbol in "from sklearn.preprocessing"
Execution halted
```