Name: Xian Hui B. Cheng
Section: BSIT 2-5

ACTIVITY 2

1. Create a R programming that can use a different operation using **concatenate**.

   a. **Arithmetic Operator** ***(let num1 = 24, num2 = 30)***
      1. Addition
      2. Subtraction
      3. Multiplication
      4. Division
      5. Modulus
      6. Integer Division
      7. Exponent

   Using concatenate method for operations:

```
> num1 <- 24
> num2 <- 30
[1] "Addition:" "54"
> sub <- c("Subtraction:", num1 - num2)
> sub
[1] "Subtraction:" "-6"
> multiply <- c("Multiplication:", num1 * num2)
> multiply
[1] "Multiplication:" "720"
> div <- c("Division:", num1 / num2)
> div
[1] "Division:" "0.8"
> modulus <- c("Modulus:", num1 %% num2)
> modulus
[1] "Modulus:" "24"
> int_division <- c("Integer Division:", num1 %/% num2)
> int_division
[1] "Integer Division:" "0"
> exponent <- c("Exponent:", num1 ^ num2)
> exponent
[1] "Exponent:"            "2.54880876153761e+41"
```

   b. **Relational Operator** ***(let num1 = 56, num2 = 45)***
      1. Less Than
      2. Less than or equal to

3. Greater than
4. Greater or equal to
5. Equal to
6. Not Equal to

```
> num1 <- 56
> num2 <- 45
```

```
> relational <- c(
+     num1 < num2,  # Less than
+     num1 <= num2, # Less than or equal to
+     num1 > num2,  # Greater than
+     num1 >= num2, # Greater or equal to
+     num1 == num2, # Equal to
+     num1 != num2 # Not equal to
+ )
> relational
[1] FALSE FALSE  TRUE  TRUE FALSE  TRUE
```

c. Logical Operation

1. *Logical not*

```
> logical_not<- c("Logical Not: ", !TRUE)
> logical_not
[1] "Logical Not: " "FALSE"
```

Explanation: ! operator negates the current value. Since the negation of TRUE is FALSE, it will return FALSE.

2. *Element-wise logical AND*

```
> x <- c(TRUE, FALSE, TRUE, FALSE)
> y <- c(TRUE, TRUE, FALSE, TRUE)
> result <- x & y
> result
[1]  TRUE FALSE FALSE FALSE
```

Explanation: The element wise logical AND will compare bit by bit

- TRUE & TRUE results in TRUE

- FALSE & TRUE results in FALSE
- TRUE & FALSE results in FALSE
- FALSE & FALSE results in FALSE

3. ***Logical AND***

Note: The Logical AND operator only works with single values, unlike the Element-wise logical AND where you can use concatenated values / entire vectors

```
> logical_and <- (10 < 5) && (!FALSE) && (210 > 20)
> logical_and
[1] FALSE
```

Explanation:

It will first evaluate (10 < 5) which returns FALSE. So it will immediately stop evaluating and return FALSE.

4. ***Element-wise logical OR***

```
> x <- c(TRUE, FALSE, FALSE, TRUE)
> y <- c(TRUE, TRUE, FALSE, FALSE)
> element_wise_or <- x | y
> element_wise_or
[1]  TRUE  TRUE FALSE  TRUE
```

Explanation: The element wise logical OR will compare bit by bit

- TRUE | TRUE results in TRUE
- FALSE | TRUE results in TRUE
- FALSE | FALSE results in FALSE
- TRUE | FALSE results in TRUE

5. ***Logical OR***

Note: Like Logical AND, the Logical OR operator only works with single values instead of the entire vectors that is concatenated

```
> logical_or <- (12 == 12) || (10 > 201) || (!TRUE)
> logical_or
[1] TRUE
```

Explanation:

It will first evaluate (12 == 12) which returns TRUE. So it will immediately stop evaluating and return TRUE. It will not check the remaining even if (10 > 201) or !TRUE results to FALSE.

BONUS:

Order of Precedence (Highest to Lowest) : ! → & → && → | → ||

```
> # Complex Version
> a<- 21
> b <- 15
> logical <- (a < b) || (!a) || (TRUE && FALSE) || (TRUE & FALSE) && !b
> logical
[1] FALSE
```

D. Combination Let num1 = 76, num2 = 56, num3 = 74, num4 = 43, num5 = False , num6 = True num7 = True

```
> num1 <- 76
> num2 <- 56
> num3 <- 74
> num4 <- 43
> num5 <- FALSE
> num6 <- TRUE
> num7 <- TRUE
```

1. (num1 > num2) || (!num6) || (!num5) || (num2 != num3)

```
> result <- (num1 > num2) || (!num6) || (!num5) || (num2 != num3)
> result
[1] TRUE
```

Explanation:

Since we used logical OR, it will only evaluate the first expression. Since the first depression is TRUE, it will automatically return TRUE

Since (num1 > num2) evaluates to TRUE, the || operator stops further evaluation, and the result of the entire expression is TRUE. The remaining conditions ((!num6), (!num5), (num2 != num3)) are not evaluated.

2. (num1 > num4) || (!num7) || (!num4) || (num2 <= num3)

```
> result <- (num1 > num4) || (!num7) || (!num4) || (num2 <= num3)
> result
[1] TRUE
```

Explanation:

The usage of logical OR only evaluates (num1 > num4). Since 76 > 43 is TRUE then it will immediately return TRUE.

3. (num5 > num4) || (!num1) || (!num6) || (num1 >= num5)

```
> result <- (num5 > num4) || (!num1) || (!num6) || (num1 >= num5)
> result
[1] TRUE
```

Explanation:

- The usage of logical OR evaluates (num5 > num4). Since FALSE > 43 is FALSE because FALSE is evaluated as 0 in numerical comparison.
- It will move on to the next expression (!num1) which returns FALSE.
- Then it will move to evaluating (!num6) which returns TRUE then it will stop evaluating and immediately return TRUE.