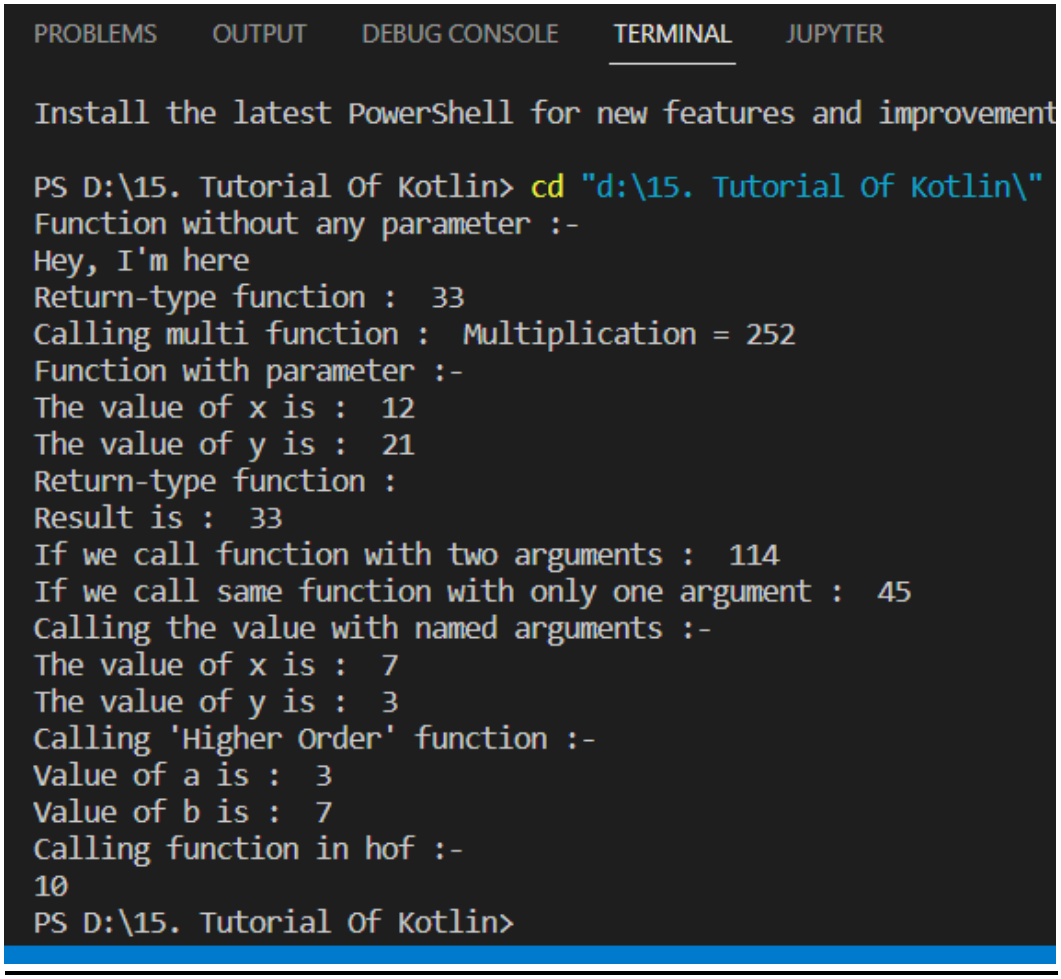
INDEX :

Function :

```
fun disp1(){

    println("Hey, I'm here")

}


fun disp2() : Int{   // return type 'Int'

    val a = 12

    val b = 21

    return(a + b)

}


fun multi() : String {

    val a = 12

    val b = 21

    return ("Multiplication = ${ a * b }")

}


fun pFun1(x:Int, y:Int){   // return type function with arguments.

    println("The value of x is :  $x")

    println("The value of y is :  $y")

}


fun pFun2(x:Int, y:Int) : Int{

    return (x + y)

}


fun deft(x:Int, y:Int = 33) : Int{

    return (x + y)

}


fun hof(a:Int, b:Int, callback:(Int, Int) -> Int){

    println("Value of a is :  $a")

    println("Value of b is :  $b")

    println("Calling function in hof :-")

    println(callback(a, b))
```

```
}


fun main() {
    // Function -> Kotlin functions are declared using 'fun' keyword.
    // Function without any parameter :-
    println("Function without any parameter :- ")
    disp1()

    println("Return-type function :  ${disp2()}")

    println("Calling multi function :  ${multi()}")

    // Function with parameter :-
    println("Function with parameter :-")
    pFun1(12, 21)

    println("Return-type function :")
    val res : Int = pFun2(12, 21)
    println("Result is :  $res")

    // Function with Default arguments :-
    println("If we call function with two arguments :  ${deft(12, 102)}")
    println("If we call same function with only one argument :  ${deft(12)}")

    // Functin with named arguments :-
    println("Calling the value with named arguments :-")
    pFun1(y=3, x=7)

    // Higher order function :-
    println("Calling 'Higher Order' function :-")
    hof(3, 7, :: pFun2)

}
```

Output :-


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
Install the latest PowerShell for new features and improvement
PS D:\15. Tutorial Of Kotlin> cd "d:\15. Tutorial Of Kotlin\"
Function without any parameter :-
Hey, I'm here
Return-type function : 33
Calling multi function : Multiplication = 252
Function with parameter :-
The value of x is : 12
The value of y is : 21
Return-type function :
Result is : 33
If we call function with two arguments : 114
If we call same function with only one argument : 45
Calling the value with named arguments :-
The value of x is : 7
The value of y is : 3
Calling 'Higher Order' function :-
Value of a is : 3
Value of b is : 7
Calling function in hof :-
10
PS D:\15. Tutorial Of Kotlin>

Lambda Expression :

```
fun add(a:Int, b:Int) : Int{
    return (a+b)
}

fun hof(a:Int, b:Int, callback:(Int, Int) -> Int){
    println(callback(a, b))
}

fun main() {
    //        **** Lambda Expression ****
    val add = {a:Int, b:Int -> a+b}
    println("This value comes from 'Lambda' Expression :  ${add(10,20)}")

    // Another way to write lambda expression :-
    val sum : (Int, Int) -> Int = {a, b -> a+b}
    println("This value also comes from 'Lambda' Expression :  ${sum(12, 21)}")

    println("Calling 'Higher Order Function'")
    hof(10, 20, ::add)  // Instead of writing this we can use lambda expression, then we don't need
    // 'add' function.

    println("Calling 'hof' and showing the use of 'Lambda' expression :-")
    hof(12, 21, {a:Int, b:Int -> a+b}) // -> This is by using 'Lambda' expression, which is easier.
    // -> Here we don't need to call add function as we use lambda expression for same.

    // We can also write above expression as follow :-
    hof(23, 56){a:Int, b:Int -> a+b} // This is recommended form to use.

    //                **** Anonymous Function ****
    val addIt = fun(a:Int, b:Int) : Int{
        return (a+b)
    }
    println("This is from 'Anonymous' function :  ${addIt(123456, 654321)}")
```

```
    val disp = fun(){
        println("Hey, everyone I'm here")
        println("This line also comes from 'Anonymos' function")
    }
    disp()

}
```

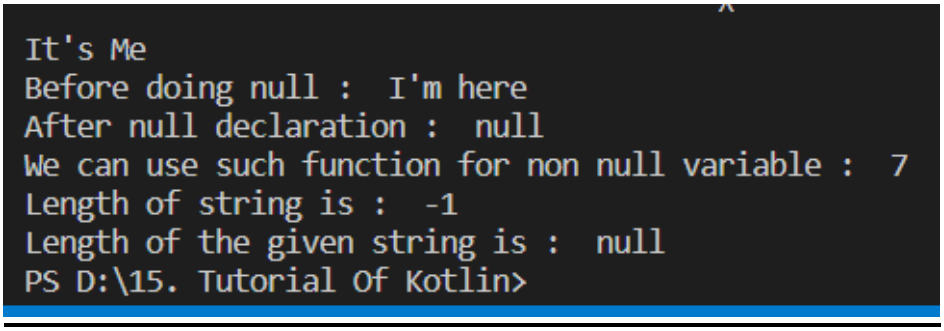
Output :-

```
PS D:\15. Tutorial Of Kotlin> cd "d:\15. Tutorial Of Kotlin\" ;
ambdaExpression.jar }
This value comes from 'Lambda' Expression :  30
This value also comes from 'Lambda' Expression :  33
Calling 'Higher Order Function'
30
Calling 'hof' and showing the use of 'Lambda' expression :-
33
79
This is from 'Anonymous' function :  777777
Hey, everyone I'm here
This line also comes from 'Anonymos' function
PS D:\15. Tutorial Of Kotlin>
```

Null Safety :

```
fun main() {

    //          **** Null Safety ****

    var name1 : String = "It's Me"

    // name1 = null     // Not allowed

    println(name1)


    var name2 : String? = "I'm here"

    println("Before doing null :  $name2")

    name2 = null

    println("After null declaration :  $name2")


    val n1ln = name1.length

    println("We can use such function for non null variable :  $n1ln")


    // val n2ln = name2.length

    // println("Some function is not allowed for the null declared variable :  $n2ln")


    // To use such function we have three methods :-

    // Option 1 :- Check null in conditions

    val nm2ln = if(name2 != null) name2.length else -1

    println("Length of string is :  $nm2ln")


    // Option 2 :- Safe Call

    val n2ln = name2?.length  // we use '?' symbol for safe call

    println("Length of the given string is :  $n2ln")


    // Option 3 :- !! Operator (not-null assertion operator)

    /* The not-null assertion operator (!!) converts any value to a non-null type and throws an

    exception if the value is null. */

    // val name2ln = name2!!.length

    // println("Length of the string is :  $name2ln")

    // This will throw exception and we have to handle it


}
```

Output :-


It's Me
Before doing null : I'm here
After null declaration : null
We can use such function for non null variable : 7
Length of string is : -1
Length of the given string is : null
PS D:\15. Tutorial Of Kotlin>

Array :

```
fun main() {

    val data = arrayOf("It's Me", "It's you", "I'm here", "Where are you", 101, 102, 'A', 'N')

    println("Printig the values in the array 'data'")

    val d1 = data[7]

    println(d1)


    // for(item in data){

    //     print(item + ", ")   // -> It doesn't work when we write so.

    // }


    for(item in data){

        print(item)

        print(", ")

    }

    println()


    // If we need array of 'Defined' variable ->

    val data1 = arrayOf<Int>(1, 2, 3, 4, 5)

    for(item in data1){

        println(item)

    }


    // For index position we have a function :-

    for(i in data.indices){

        println("$i = ${data[i]}")

    }


    // 'for-each' loop in Kotlin :-

    println("Displaying elements of the array by the use of 'for-each' loop :-")

    data.forEach{item -> println(item)}


    // Array Constructor

    val tb = Array(5, {i -> i*2})

    println("Printing the values in the array :")
```

```
    for(i in tb){

        println(i)

    }


    // Built-in method to create array :

    val roll = intArrayOf(101, 102, 103, 104)

    println("Printing the value of the array created by 'Built-in' method :-")

    for(item in roll){

        println(item)

    }


}
```

Output :-

```
PS D:\15. Tutorial Of Kotlin> cd "d:\15. Tutorial Of Kotlin\" ; if ($?) {
Printig the values in the array 'data'
N
It's Me, It's you, I'm here, Where are you, 101, 102, A, N,
1
2
3
4
5
0 = It's Me
1 = It's you
2 = I'm here
3 = Where are you
4 = 101
5 = 102
6 = A
7 = N
Displaying elements of the array by the use of 'for-each' loop :-
It's Me
It's you
I'm here
Where are you
101
102
A
N
Printing the values in the array :
0
2
4
6
8
Printing the value of the array created by 'Built-in' method :-
101
102
103
104
PS D:\15. Tutorial Of Kotlin>
```

User Input Array :

```kotlin
fun main() {

    print("Enter the size of the array here :  ")

    val num = readLine()!!.toInt()



    println("Enter the elements in the array :-")

    val ele = Array(num){readLine()!!.toInt()}



    println("Printing the elements in the array :-")

    for(el in ele){

        print(el)

        print(", ")

    }


}
```

Output :-

```
PS D:\15. Tutorial Of Kotlin> cd "d:\15.
nputArray.jar }
Enter the size of the array here :  4
Enter the elements in the array :-
1
2
3
4
Printing the elements in the array :-
1, 2, 3, 4,
PS D:\15. Tutorial Of Kotlin>
```

List :

```
fun main() {
    /*
    List -> List is an ordered collection with access to elements by indices - integer numbers that
    reflact their position. Elements can occur more than once in a list.
    */

    val data = listOf("Aman", "You", 21, 'A', 'N') // Creating the list of variables.
    // We cann't add or update any value in the above list.
    println("Printing the elements in the list 'data' :  ${data}")

    println("We can sepratelly print the elements of list :-")
    println(data[0])
    println(data[3])
    println(data[4])
    println(data.get(2))  // Another way to print element.
    println("Printing the elements by the use of 'for' loop :-")
    for(item in data){
        println(item)
    }

    println("Printing the elements in the list by use of 'for-each' loop :-")
    data.forEach{
        dt ->
        print(dt)
        print(", ")
    }
    println()

    // If we want to add or update any value further then we have to make following list.

    val names = mutableListOf<String>("Hey", "I'm", "here", "for", "you")
    println(names)

    names[0] = "Aman"   // -> here we update the value at names[0]
```

```
    names.add(5, "How are you")   // -> here we add data to the list.

    names.removeAt(1)   // -> here we remove the element of the list.

    println(names)



    // Taking input form user for list :-

    print("Enter the size of list :  ")

    val num = readLine()!!.toInt()


    println("Enter the elements in the list :-")

    val elem = List<Int>(num){readLine()!!.toInt()}


    println("Elements entered by the user is as follow :-")

    println(elem)


}
```

Output :-

```
PS D:\14. Tutorial Of Python> python -u "d:\14. Tutorial Of Python\F_Tuples.py"
Printing the elements of the tuple :   (1, 2, 3, 4, 1, 2, 3, 1, 1)
Tuple with single element :   (1,)
Printing the count of any element in the tuple :   4
Checking the idex of elements :   1
PS D:\14. Tutorial Of Python>
```

Set :

```kotlin
fun main() {

    // We cann't make any change in this set as this not a mutable set(Immutable set).

    val data = setOf("Aman", "Verma", "Its Me", "Aman", 1, 2, 1, 'A', 'N', 'V', 'A')

    println(data)


    println("This will give the size of set :  ${data.size}")


    // Defined set

    val names = setOf<String>("Hey", "I'm", "here")

    println(names)


    // For mutable set to make desired change we have to do so ->

    val data1 = mutableSetOf("Aman", "Aditya", "Shubham", 2, 23, 3, 'A', 'D', 'S')

    println("Printing the values in the set 'data1' :  ${data1}")


    println("Now we can add or remove elements in the above set")

    data1.add("Hey")

    println("Printing the elements after adding an element :  ${data1}")


    data1.remove('D')

    println("Printing the elements after removing an element :  ${data1}")


}
```

Output :-

```
PS D:\15. Tutorial Of Kotlin> cd "d:\15. Tutorial Of Kotlin\" ; if ($?) { kotlinc M_Set.kt -includ
[Aman, Verma, Its Me, 1, 2, A, N, V]
This will give the size of set :  8
[Hey, I'm, here]
Printing the values in the set 'data1' :  [Aman, Aditya, Shubham, 2, 23, 3, A, D, S]
Now we can add or remove elements in the above set
Printing the elements after adding an element :  [Aman, Aditya, Shubham, 2, 23, 3, A, D, S, Hey]
Printing the elements after removing an element :  [Aman, Aditya, Shubham, 2, 23, 3, A, S, Hey]
PS D:\15. Tutorial Of Kotlin>
```

Map :

```
fun main() {
    /*
    Map (Dictionary) -> Map is set of key-value pairs. Keys are unique, and each of them maps to
    exactly one value. The values can be duplicates. Maps are useful for storing logical connections
    between objects.
    Ex :- An employee ID and their position.
    Syntax for map : val data = mapOf(key to value)
    */

    val data = mapOf(1 to "Aman", "Key2" to 12, "Key3" to "Verma")
    println("Printing the value in the map :  ${data}")

    // If we want to access particular data :-
    println(data[1])
    println(data["Key2"])
    println(data["Key3"])
    println(data.get(1))   // Another way to get value
    println(data.get("Key2"))

    // If we only want to have 'key' values :-
    println("Keys are as follow :-")
    println(data.keys)

    // If we only want ot have 'values' :-
    println("'Values' are as follow :-")
    println(data.values)

    // Syntax of defined data-type of map :
    // val data = mapOf<String, String>(key to value) -> If both key & values in String.

    // Printing the values in the map by the use of for loop :-
    println("Printing the values in the map by the use of for loop")
    for(item in data.keys){
        println("$item = ${data[item]}")
```

```
    }


    // For mutable map through which we make changes in the map :-
    val mapData = mutableMapOf(1 to "Aman", "Key2" to 12, "Key3" to "Verma", 3 to "Remove It")
    println("Printing the mutable map :  ${mapData}")


    mapData["Key2"] = "Hello"    // update the value in the mapData
    mapData.put(1, "Aman V.")    // another way to update the mapData


    mapData.remove("Key3")       // remove some data
    mapData.keys.remove("Key2")  // another way to remove some data, these both are based on keys.


    mapData.values.remove("Remove It")  // removing data based on values.


    mapData[4] = "How are you"   // to add data in mapData, Note :- keys are unique.
    println("Printing the mutable map :  ${mapData}")


}
```

Output :-

```
PS D:\15. Tutorial Of Kotlin> cd "d:\15. Tutorial Of Kotlin\" ; if ($?)
Printing the value in the map :  {1=Aman, Key2=12, Key3=Verma}
Aman
12
Verma
Aman
12
Keys are as follow :-
[1, Key2, Key3]
'Values' are as follow :-
[Aman, 12, Verma]
Printing the values in the map by the use of for loop
1 = Aman
Key2 = 12
Key3 = Verma
Printing the mutable map :  {1=Aman, Key2=12, Key3=Verma, 3=Remove It}
Printing the mutable map :  {1=Aman V., 4=How are you}
PS D:\15. Tutorial Of Kotlin>
```