**Exploring Data Classes in Kotlin**

**Objective:**

The objective of this lab exercise is to understand the concepts and features of data classes in Kotlin using the domain of Formula 1 racing.

1. Open Android Studio and create a new Android project.
2. Create a new Kotlin file called "Driver.kt" and define a data class called "Driver" with the following properties:

* name: String
* team: String
* country: String
* number: Int

In the MainActivity file, initialize multiple instances of the Driver class with different values, representing different Formula 1 drivers.

Use the properties and methods provided by data classes to perform the following tasks:

**Task 1: Printing Data**

Print the details of each driver object using logcat

**Task 2: Equality Check**

Compare two driver objects for equality using the "==" operator.

Compare two driver objects for inequality using the "!=" operator.

Print the results of both comparisons.

**Task 3: Copying Data**

Create a copy of a driver object and assign it to a new variable.

Modify some properties of the copied object.

Print the original and modified objects to observe the differences.

**Task 4: Destructuring Declarations**

Destructure a driver object into separate variables: name, team, country, and number.

Print the values of the destructured variables.

**Task 5: toString() Method**

Display the string representation of each driver object using the default toString() method.

Run the Android application and observe the output in the logcat or on the device's screen.

**Exploring Enums in Kotlin**

**Objective:**

The objective of this lab exercise is to understand the concepts and features of enums in Kotlin using the domain of Formula 1 racing in an Android Studio project.

**Instructions:**

1. Open Android Studio and create a new Android project.
2. Create a new Kotlin file called "MainActivity.kt" and define a MainActivity class.
3. Inside the MainActivity class, define an enum class called "Team" with the following teams:

Mercedes

Red Bull Racing

Ferrari

McLaren

Aston Martin

Alpine

AlphaTauri

Alfa Romeo

Haas

Williams

1. Inside the MainActivity class, define a function called "printTeamDetails" with the following signature:

private fun printTeamDetails(team: Team) {

// Code for printing team details

}

1. Inside the function body of "printTeamDetails", use control flow statements to perform the following tasks:

**Task 1: Print Team Details**

Print the details of a team using the provided parameter.

**Task 2: Switch Statement**

Implement a switch statement to check the team and print the corresponding message.

**Task 3: Call the method**

In the MainActivity's onCreate() method, call the printTeamDetails function with a sample team.

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

val team = Team.MERCEDES

printTeamDetails(team)

}

**Algorithms Using Kotlin Exercise 2**

Given an array of buildings and a direction that all the buildings face, return an array of the indices of the buildings that can see the sunset.

A building can see the sunset if it's strictly taller than all of the buildings that come after it in the direction that it faces.

The input array named ***buildings*** contains positive, non-zero integers representing the heights of the buildings. A building at index ***i*** thus has a height denoted by ***buildings[i]***

All of the buildings face the same direction, and this direction is either east or west, denoted by the input string named ***direction***, which will always be equal to either "EAST" or "WEST"

In relation to the input array, you can interpret these directions as right for east and left for west.

Important note: the indices in the ouput array should be sorted in ascending order.

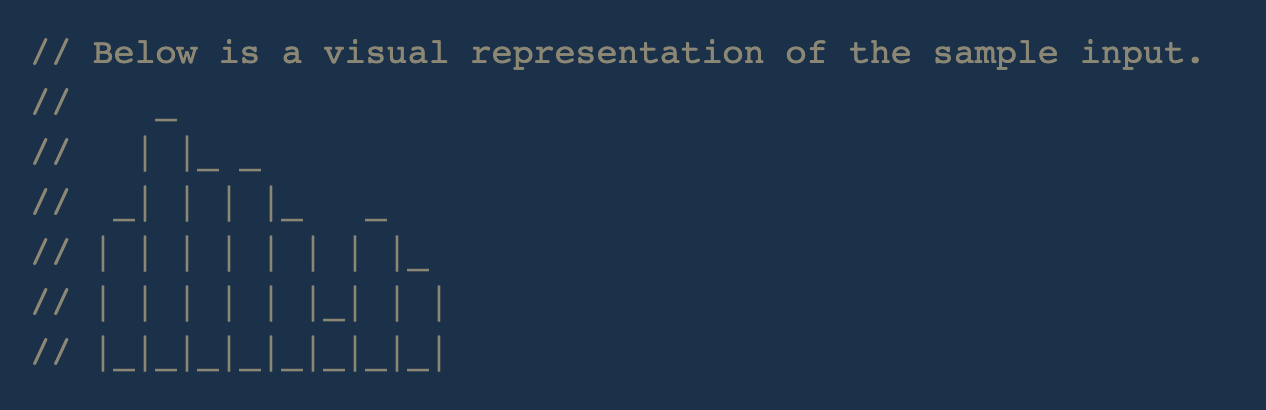
**Sample Input:**

buildings = [3, 5, 4, 4, 3, 1, 3, 2]

direction = "EAST"

**Sample output**

= [1, 3, 6, 7]



**Sample Input:**

buildings = [3, 5, 4, 4, 3, 1, 3, 2]

direction = "WEST"

**Sample Output**

= [0, 1]

// The buildings are the same as in the first sample

// input, but their direction is reversed.

**Exercise 3**

**First Non-Repeating Character**

Write a function that takes in a string of lowercase English-alphabet letters and returns the index of the string's first non-repeating character. The first non-repeating character is the first character in a string that occurs only once.

If the input string doesn't have any non-repeating characters, your function should return -1

**Sample Input**

string = "abcdcaf"

**Sample Output**

1 // The first non-repeating character is "b" and is found at index 1.