**Pair Programming Logic Lab 5**

**In the following exercise, you will be using JS to solve logical problems in pairs.**

* Create a file “Name1Name2Lab5.js”
* For each problem create a function and name it **as the problem states**
* All functions should be arrow functions

**Problem 1:🔑x5**

Write a function **“getProperties”** that take an object as argument and return its properties in array form

**Sample Input** **Sample Output**

{ a: 1, b: true , c: “I don’t know” } [a, b, c]

**Problem 2:🔑x5**

Write a function **“isPlainObject”** that verifies an argument is a plain object, not anarray or null. It will return true if object is plain, false otherwise.

**Sample Input** 7 **Sample Output**

{ a: 1 } true

[1, 2, 3] false false

**Problem 3:🔑x5**

Write a function **“modifyObject”** that :  
1) adds a new property to your object called **isObject** and set it to true or false after checking if it’s an object.  
2) adds a new property to your object which is a function called **getPropertiesNb** that returns the number of objects properties.

**Sample Input** **Sample Output**

{ a: 1, c: “I don’t know” } { a:1, c: "I don’t know", isObject:**false**, getPropertiesNb: {...} }

**Problem 4:🔑x5**

Write a function “**makePairs**” that returns a deep array like [[key, value]]

**Sample Input** **Sample Output**{ a: 1, b: 2 } [['a', 1], ['b', 2]]

**Problem 5:🔑x5**

Write a function “**without**” that take an object and propertie(s) as arguments and returns a new object without provided properties.

**Sample Input** **Sample Output**{ a: 1, b: 2 } 'b' { a: 1 }

**Problem 6:🔑x5**

Write a function “**isEmpty**” that makes a shallow check if object is empty.

**Sample Input** **Sample Output**{} true  
{ a: undefined } true  
{ a: 1 } false  
{ a: 1, b: undefined } false

**Problem 7:🔑x5**

Write a function “**isEqual**” that makes a shallow compare of two objects. It returns true if objects are identical and false otherwise.

**Sample Input** **Sample Output**

{ a: 1, b: 1 } { a: 1, b: 2 } false

**Problem 8:🔑x5**

Write a function “**intersection**” that finds shallow intersections of objects and return it in a new object.

**Sample Input** **Sample Output**

{ a: 1, b: 2 } { c: 1, b: 2 } { b: 2 }