CS 315 Homework 1

Associative Arrays in Dart, Javascript, Lua, PHP, Python, Ruby, and Rust

Ant Duru

ID: 21704108 Section: 02

This report is divided into seven parts, one for each programming language that includes the code, the output and the explanations of what my code does. In code parts, I added comments that state which operation is being done and the number of that operation (1-Initializing the array, 2- Getting an element by key etc.) to refer to them later in my report with their numbers.

DART

Code:

```
void main(){
   // Initializing the hash
   Map myMap = {
                  'first': 1,
                  'second': 2,
                  'third': 3,
                  'fourth': 4,
                  'fifth': 5};
   print("-----------1-Associative Array Has Been
Initialized.-----------");

   // Getting value by a given key
   print("-----------2-Getting the value of an element.-----------");
   var myValue = myMap["first"];
   print("value with the key \"first\" is:  $myValue ");

   // Adding an element
   print("-----------3-Adding an element.-----------");
   myMap["sixth"] = 6;
   print("An element with key \"sixth\" and value 6 has been added");

   // Removing an element
   print("-----------4-Removing an element.-----------");
   myMap.remove('third');
   print("The element with key \"third\" has been removed");

   // Modifying an element
   print("-----------5-Modifying an element.-----------");
   var before = myMap['fourth'];
   print("Before modify, the value with key \"fourth\" was: $before");
   myMap["fourth"] = 7;
   var after = myMap['fourth'];
   print("After modify, the value with key \"fourth\" is: $after");

   // Search for existence of a key
   print("-----------6-Searching for existence of a key.-----------");
   if(myMap["first"] != null){
       var fValue = myMap['first'];
       print("the key \"first\" is in the hash with value: $fValue ");
```

```
    }
    else{
        print("the key \"first\" is not in the hash");
    }

    // Search for existence of a value
    print("-----------7-Searching for existence of a
value.-----------");
    var e = myMap.keys.firstWhere((k) => myMap[k] == 2, orElse: () =>
null);

    if(e == null)
    {
        print("the value 2 is not in the hash");
    }
    else{
        print("the value 2 is in the hash");
    }

    // Loop through the hash and print pairs
    print("-----------8-Iterating through the hash and printing
pairs.-----------");
    void foo(k, v){
        print("key: $k , value: $v");
    }

    myMap.forEach((k, v) {
     foo(k,v);
     });
}
```

Output:

```
----------1-Associative Array Has Been Initialized.----------
----------2-Getting the value of an element.----------
value with the key "first" is:  1
----------3-Adding an element.----------
An element with key "sixth" and value 6 has been added
----------4-Removing an element.----------
The element with key "third" has been removed
----------5-Modifying an element.----------
```

```
Before modify, the value with key "fourth" was: 4
After modify, the value with key "fourth" is: 7
-----------6-Searching for existence of a key.-----------
the key "first" is in the hash with value: 1
-----------7-Searching for existence of a value.-----------
the value 2 is in the hash
-----------8-Iterating through the hash and printing pairs.-----------
key: first , value: 1
key: second , value: 2
key: fourth , value: 7
key: fifth , value: 5
key: sixth , value: 6
```

Explanation:

1- To initialize the associative array, I have created a Map using Map literal and on the left-hand side, I entered keys (as strings) and values (as integers), separated with colons. Pairs are separated with commas and all of the pairs are in curly brackets.

2- To get the value with key “first” in Dart, I have created a variable and the value of the variable is expressed as myMap[“first”]. This expression is really similar to getting values from an array in Java. However, instead of index, the key whose value is wanted is written.

3- An element in a map needs two properties: key and value. To add a new element, the notation to get values(myMap[key]) is used but on the right-hand side. Since I wanted to add an element with key “sixth” and value 6, I assigned 6 to my expression myMap[“sixth]. It creates a new pair with key “sixth” and then assigns its value as 6, which completes the pair.

4- To remove an element, a built-in function called remove() is used. It, simply, deletes the key-value pair from the map. This function, actually, takes the key as parameter and returns the removed key but I did not really need to see the key again, since I did not take the key from the user etc.

5- To modify an element is really similar to adding a new element. The only difference in the notation is that to modify, the key should already exist. In my code, I displayed the value of the same key two times, before and after modifying.

6- To search for the existence of a key, I wrote a segment to check whether a value of a given key is null. If the value for a key is null, it means that there is no such key in the map. The program outputs whether the key is in the map or not.

7- To search for the existence of a value, I defined a variable e. e is the result of an expression, which uses a property of the Map class, keys. keys is an Iterable. A built-in function firstWhere() is applied to keys. firstWhere function is to find a first appearance of something in an Iterable. What I was looking for is whether a pair in myMap has value 2. So, I looked in firstWhere whether a key has the value 2. If there is such a key, firstWhere will return the key, else, it will return null. Then I checked whether e is null or not. Finally, the program displays the appropriate message containing information about whether 2 is in the map or not.

8- The foo() function has two parameters: k,v which will be the key and value of the map for each iteration. The function does not return anything and only prints the key value pair. The function is called in a for loop built with the forEach, which gets the collection of pairs in myMap. foo() function is called for every pair.

JAVASCRIPT

Code:

```
// Initializing the associative array
var myMap = {"first": 1, "second": 2, "third": 3, "fourth":4,
"fifth":5}
console.log("1-Associative array has been initialized.");

// Getting the value given key
console.log("2-Getting value given a key.");
var myValue = myMap["second"];
console.log("the value for the key \"second\" is: ");
console.log(myValue);

// Add a new element
console.log("3-Adding a new element.");
myMap["sixth"] = 6;
console.log("A new element with key \"sixth\" and value 6 has been
added.");

// Removing an element
console.log("4-Removing an element.");
delete myMap["third"];
console.log("The element with key \"third\" has been removed.");

// Modifying an element
console.log("5-Modifying an element.");
```

```
console.log("Before Modify, the element with key \"fourth\" had value:
");
console.log(myMap["fourth"]);
myMap["fourth"] = 8;
console.log("After Modify, the element with key \"fourth\" has value:
");
console.log(myMap["fourth"]);

// Search existence of key
console.log("6-Search existence of key \"first\".");
if (myMap["first"] !== null)
{
    console.log("the key \"first\" is in the map with value: ");
    console.log(myMap["first"]);
}
else
{
    console.log("the key \"first\" is not in the map");
}

// Search existence of value
console.log("7-Search existence of value 8");
var bool = false;
var k;
for(var key in myMap) {
    if(myMap.hasOwnProperty(key)) {
        if(myMap[key] == 8)
        {
            bool = true;
            k = key;
            break;
        }
    }
}
if(bool === true)
{
    console.log("the value 8 exists in the map with key: ");
    console.log(k);
}
else
{
    console.log("the value 8 does not exist in the map");
}
```

```
function foo( k, v){
    console.log("key: ");
    console.log(k);
    console.log("value: ");
    console.log(v);
    console.log("-------");
}

// Loop to iterate through map and print pairs
console.log("------8-LOOP TO ITERATE-------");
for (let k in myMap)
{
    var value = myMap[k];
    foo(k, value);
}
```

OUTPUT:

1-Associative array has been initialized.
3-Getting value given a key.
the value for the key "second" is:
2
3-Adding a new element.
A new element with key "sixth" and value 6 has been added.
4-Removing an element.
The element with key "third" has been removed.
5-Modifying an element.
Before Modify, the element with key "fourth" had value:
4
After Modify, the element with key "fourth" has value:
8
6-Search existence of key "first".
the key "first" is in the map with value:
1
7-Search existence of value 8
the value 8 exists in the map with key:
fourth
------8-LOOP TO ITERATE-------
key:
first
value:
1

```
-------
key:
second
value:
2
-------
key:
fourth
value:
8
-------
key:
fifth
value:
5
-------
key:
sixth
value:
6
-------
```

Explanation:

1- To initialize an associative array, I have created a variable called myMap and initialized it to pairs of strings (keys) and integers (values) separated by colons. Pairs are separated by commas.

2- To get the value of an element given key (in this case I want the key to be "second"), I have created a variable called myValue and equalized it to myMap["second"]. This is also similar to what I have done in Dart language. This expression is similar to get value in an array by index but instead of index, one should use the key.

3- Adding a new element to the map in JavaScript is the same as Dart. I wanted to add a new element with key "sixth" and value 6.

4- To delete an element, there is a keyword **`delete`**. I wanted to delete the element with the key "third". So, the expression is `delete myMap["third"];`

5- Modifying an element is also the same with Dart language. It is the same syntax with adding a new element, but the key should already exist.

6- To search existence for a key, I checked whether a value with a given key (in this case, "first") is null or not. If there is no key as "first" the value should be null. If the value is not null, the key exists.

7- To search for the existence of a value, I iterate through all keys, using a built in function hasOwnProperty() that returns the property passed as an argument, if exists. I passed keys as parameters and checked whether a key has value 8. If there is a key that has the value 8, a variable I formerly declared as false will be true.

8- There is a simple for loop iterating through the keys of myMap. In every iteration, it declares a value which builds the pair with the key and calls the foo function. foo function simply prints the key, value pair.

Lua

Code:

```
-- my function definition to print pairs
function foo(k, v)
    print("key: ", k, ", value: ", v)
end

-- Initializing the table
local myTable = { ["first"] = 1, ["second"] = 2, ["third"] = 3,
["fourth"] = 4}
print("--------1-TABLE HAS BEEN INITIALIZED---------")

-- Getting value for a given key
print("--------2-GETTING VALUE BY KEY---------")
local myValue = myTable["first"]
print("the value for the key \"first\" is: ", myValue)

-- Adding an element
print("--------3-ADDING AN ELEMENT---------")
myTable["sixth"] = 6
print("An element with key \"sixth\" and value 6 has been added.")

-- Removing an element
print("--------4-REMOVING AN ELEMENT---------")
myTable["third"] = nil
print("The element with key \"third\" has been removed.")

-- Modifying an element
print("--------5-MODIFYING AN ELEMENT---------")
print("Before Modify, the value with key \"fourth\" is: ",
myTable["fourth"])
```

```
myTable["fourth"] = 5
print("After modify, the value with key \"fourth\" is: ",
myTable["fourth"])

-- Search for existence of key
print("--------6-LOOKING IF KEY \"second\" exists---------")
if(myTable["second"] ~= nil) then
    print("The key \"second\" exists in the table with value: ",
myTable["second"])
else
    print("The key \"second\" does not exist in the table")
end


-- Search for existence of value,
print("--------7-LOOKING IF VALUE 56 exists---------")
local bool = false
for k, v in pairs(myTable) do
    if v == '56' then
        print("56 is in the table with key: ", k)
        bool = true
    end
end
if bool ~= true then
    print("56 is not in the table.")
end

-- Loop
print("--------8-LOOP THROUGH THE TABLE AND PRINT THE PAIRS---------")
for k,v in pairs(myTable) do foo(k,v) end
```

Output:

```
--------1-TABLE HAS BEEN INITIALIZED---------
--------2-GETTING VALUE BY KEY---------
the value for the key "first" is:       1
--------3-ADDING AN ELEMENT---------
An element with key "sixth" and value 6 has been added.
--------4-REMOVING AN ELEMENT---------
The element with key "third" has been removed.
--------5-MODIFYING AN ELEMENT---------
```

```
Before Modify, the value with key "fourth" is:          4
After modify, the value with key "fourth" is: 5
--------6-LOOKING IF KEY "second" exists---------
The key "second" exists in the table with value:       2
--------7-LOOKING IF VALUE 56 exists---------
56 is not in the table.
--------8-LOOP THROUGH THE TABLE AND PRINT THE PAIRS---------
key:    sixth    , value:          6
key:    second , value:          2
key:    fourth   , value:          5
key:    first      , value:          1
```

Explanation:

1- To initialize an associative array in Lua, I have created a variable called myTable, using the appropriate syntax to have an associative array.

2- To get the value given a key, I used the same notation as in Dart and JS.

3- Adding a new element is also the same as in Dart and JS.

4- To remove an element, I equalized a value given key (in this case "third") to nil.

5- Modifying an element is also the same as in Dart and JS

6- To search for the existence of a given key, there is an if condition to check whether a value given a key is nil or not. If the value for a given key is nil, it means that the key does not exist.

7- To search for the existence of a value, I, with a for loop, iterate through pairs in myTable using a built-in function called pairs() that takes myTable as a parameter and returns the keys and values and check if a value is equal to the value I am looking for (in this case, 56).

8- Similar to the previous operation, the program iterates through the key, value pairs using the pairs built-in function and calls foo function for each pair. foo function simply prints the key-value pair.

PHP

Code:

```php
<?php
// Defining the function to use later to print key,value pairs
function foo($arr, $key){
    if(array_key_exists($key, $arr)){
        echo $key, "    ", $arr[$key];
        echo "\n";
    }

}



// Initializing the associative array
$myArr = array(
            "first" => 10,
            "second" => 65,
            "third" => 4,
            "fourth" => 0,
            "fifth" => 77);
echo "---------------1-ASSOCIATIVE ARRAY HAS BEEN
INITIALIZED.----------------";



// Getting and printing a value by its key
echo "\n";
echo "---------------2-GETTING THE VALUE OF KEY
\"third\".----------------";
$myValue = $myArr["third"];
echo "\n";
echo "key: third, value: " , $myValue;



// Adding a new element with key "sixth" and value 6
echo "\n";
echo "---------------3-ADDING A NEW ELEMENT.----------------";
$myArr["sixth"] = 6;
```

```php
echo "\n";
echo "A new element with key \"sixth\" has been added, its value is: ",
$myArr["sixth"];



// Removing an element
echo "\n";
echo "---------------4-REMOVING AN ELEMENT.----------------";
echo "\n";
unset($myArr["second"]);
echo "The element with key \"second\" has been removed.";



// Modifying the value of an element
echo "\n";
echo "---------------5-MODIFYING THE VALUE OF AN
ELEMENT.----------------";
echo "\n";
echo "The current value of the key \"first\" is: ", $myArr["first"];
$myArr["first"] = 56;
echo "\n";
echo "After the modify, the value of the key \"first\" is: ",
$myArr["first"];



// Search for existence of a key
echo "\n";
echo "---------------6-SEARCHING EXISTENCE OF KEY
\"second\".----------------";
echo "\n";
if( array_key_exists("second", $myArr)){
    echo "the key \"second\" is in the associative array with value: ",
$myArr["second"];
}
else{
    echo "the key \"second\" is not in the associative array";
}
```

```php
// Search for existence of a value
echo "\n";
echo "---------------7-SEARCHING EXISTENCE OF VALUE
77.----------------";
echo "\n";
if( array_search(77, $myArr) != FALSE){
    echo "77 is in the associative array with key: ", array_search(77,
$myArr);
}
```

```php
// Loop through the array and printing the pairs using my function
echo "\n";
echo "---------------8-PRINTING PAIRS.----------------";
echo "\n";
foreach($myArr as $key => $value){
    foo($myArr, $key);
}
?>
```

Output:

```
---------------1-ASSOCIATIVE ARRAY HAS BEEN INITIALIZED.----------------
---------------2-GETTING THE VALUE OF KEY "third".----------------
key: third, value: 4
---------------3-ADDING A NEW ELEMENT.----------------
A new element with key "sixth" has been added, its value is: 6
---------------4-REMOVING AN ELEMENT.----------------
The element with key "second" has been removed.
---------------5-MODIFYING THE VALUE OF AN ELEMENT.----------------
The current value of the key "first" is: 10
After the modify, the value of the key "first" is: 56
---------------6-SEARCHING EXISTENCE OF KEY "second".----------------
the key "second" is not in the associative array
---------------7-SEARCHING EXISTENCE OF VALUE 77.----------------
77 is in the associative array with key: fifth
---------------8-PRINTING PAIRS.----------------
first    56
```

```
third    4
fourth    0
fifth    77
sixth    6
```

Explanation:

1- To initialize an associative array, I used the keyword `array` and used the appropriate syntax to declare key and value pairs.

2- Getting the value given a key is the same as Dart, JS and Lua.

3- Adding a new element is the same as Dart, JS and Lua.

4- To remove an element, a built-in function called unset() is used. As the parameter, `$myArr["second"]` is passed. (to remove the element with key "second")

5- Modifying an element is the same as Dart, JS and Lua.

6- To search for the existence of a key, a built-in function called array_key_exists() is used. As parameters, the key that is searched and the associative array is passed. This function returns either true or false. Appropriate message according to the returning value is displayed.

7- To search for the existence of a value, another built-in function called array_search() is used. As parameters, the value that is searched and the associative array is passed. This function also returns either true or false. An appropriate message according to the returning value is displayed.

8- To iterate through the associative array, foreach loop is used. Therefore, keys and values of each pair are obtained. In every iteration foo function is called. foo function simply prints the key-value pairs.

Python

Code:

```
# Implementing my function to use it later to print the pairs
def foo(k, d):
    print(k, d[k])


# Initializing the dictionary
my_dict = {
        "first": 10,
        "second": 65,
        "third": 4,
        "fourth": 0,
        "fifth": 77 }
print("--------1-DICTIONARY HAS BEEN INITIALIZED---------")

# Getting the value for a given key
print("--------2-GETTING THE VALUE WITH \"third\" KEY---------")
my_value = my_dict["third"]
print("key: third, value: ", my_value)

# Adding element to the dictionary with key "sixth" and value 6
print("--------3-ADDING AN ELEMENT---------")
my_dict["sixth"] = 6
print("value ", my_dict["sixth"], " is added to the dictionary with key
\"sixth\"")

# Removing element from the dictionary
print("--------4-REMOVING THE ELEMENT WITH KEY \"fourth\"---------")
my_dict.pop("second")
print("element with key \"second\" has been removed")

# Modifying an element
print("--------5-MODIFYING THE ELEMENT WITH KEY \"third\"---------")
print("current value of element with key \"first\" is: ",
my_dict["first"])
my_dict["first"] = 56
print("After modify value of element with key \"first\" is: ",
my_dict["first"])
```

```
# Searching for the existence of a key
print("--------6-SEARCHING FOR THE EXISTENCE OF A KEY---------")
if (my_dict.get("second") != None):
    print("the key \"second\" is in the dictionary with value: ",
my_dict["second"])
else:
    print("the key \"second\" is not in the dictionary")


# Searching for the existence of a value
print("--------7-SEARCHING FOR THE EXISTENCE OF A VALUE---------")
if(list(my_dict.keys())[list(my_dict.values()).index(77)] != None):
    print("The value 77 is in the dictionary with key: ",
list(my_dict.keys())[list(my_dict.values()).index(77)])
else:
    print("The value 77 is not in the dictionary")

# Loop to print pairs
print("--------8-PRINTING PAIRS---------")
for k in my_dict:
    foo(k, my_dict)
```

Output:

```
--------1-DICTIONARY HAS BEEN INITIALIZED---------
--------2-GETTING THE VALUE WITH "third" KEY---------
key: third, value:  4
--------3-ADDING AN ELEMENT---------
value  6  is added to the dictionary with key "sixth"
--------4-REMOVING THE ELEMENT WITH KEY "fourth"---------
element with key "second" has been removed
--------5-MODIFYING THE ELEMENT WITH KEY "third"---------
current value of element with key "first" is:  10
After modify value of element with key "first" is:  56
--------6-SEARCHING FOR THE EXISTENCE OF A KEY---------
the key "second" is not in the dictionary
--------7-SEARCHING FOR THE EXISTENCE OF A VALUE---------
The value 77 is in the dictionary with key:  fifth
--------8-PRINTING PAIRS---------
first 56
third 4
```

fourth 0
fifth 77
sixth 6

Explanation:

1- An associative array in Python is called a dictionary. A dictionary is created with appropriate syntax (strings as keys and integers as value).

2- Getting a value given key is the same as Dart, JS, Lua and PHP.

3- Adding a new element is the same as Dart, JS, Lua and PHP.

4- To remove an element from the dictionary, a built-in function pop() is called with a parameter that is the key of the element wanted to be removed.

5- Modifying an element is the same as Dart, JS, Lua and PHP.

6- To search for the existence of a key, a built-in function called get() is used. get() function takes the key as parameter and returns the value associated with that key. If there is not any value, it returns None.

7- To search for the existence of a value, two built-in functions called keys() and values() are used. If there is the value 77 in values of the keys, it returns the associated key. Else, it returns None.

8- To print pairs, a for loop is used. For loop iterates through keys in the dictionary. For each iteration, foo function is called. Foo function has parameters k, d; which are for key and the dictionary. Then the function prints the key-value pair for each iteration.

RUBY

Code:

```ruby
# my method to print
def foo(k, v)
    puts "key: #{k} , value: #{v}"
end


# Initializing the array
myHash = {"first" => 7, "second" => 19, "third" => 63, "fourth" => 81,
"fifth" => 109}
puts "------1-Hash has been initialized.-------"


# Getting Value for a given key
puts "------2-Getting the value of key \"first\".-------"
myValue = myHash["first"]
puts "the value with key \"first\" is: #{myHash["first"]}"

# Adding a new element
myHash["sixth"] = 6
puts "------3-Element with key \"sixth\" and value 6 is added.-------"


# Removing an element
myHash.delete("third")
puts "------4-Element with key \"third\" is deleted.-------"

# Modify the value of an element
puts "------5-Modifying value of an element.-------"
puts "The value of the element with key \"fifth\" is:
#{myHash["fifth"]}"
myHash["fifth"] = 5
puts "After modification the element with key \"fifth\" is:
#{myHash["fifth"]}"

# Search for existence of a key
puts "------6-Searching whether key \"second\" is in the hash.-------"
```

```ruby
if myHash["third"] != nil
    puts "the key \"third\" is in the hash with value:
#{myHash["third"]}"
else
    puts "the key \"third\" is not in the hash"

end


# Search for existence of a value
puts "------7-Searching whether value 81 is in the hash.-------"
if myHash.key(81) != nil
    puts "81 is in the hash with key: #{myHash.key(81)}"
else
    puts "81 is not in the hash"

end

# Loop
puts "------8-Iterating through hash and printing pairs.-------"
myHash.each do |key, value|
    foo(key, value)
end
```

Output:

```
$ruby main.rb
------1-Hash has been initialized.-------
------2-Getting the value of key "first".-------
the value with key "first" is: 7
------3-Element with key "sixth" and value 6 is added.-------
------4-Element with key "third" is deleted.-------
------5-Modifying value of an element.-------
The value of the element with key "fifth" is: 109
After modification the element with key "fifth" is: 5
------6-Searching whether key "second" is in the hash.-------
the key "third" is not in the hash
------7-Searching whether value 81 is in the hash.-------
81 is in the hash with key: fourth
------8-Iterating through hash and printing pairs.-------
key: first , value: 7
```

key: second , value: 19
key: fourth , value: 81
key: fifth , value: 5
key: sixth , value: 6

Explanation:

1- Created a hash with appropriate syntax.

2- Getting an element is the same as in Dart, JS, Lua, PHP and Python

3- Adding a new element is the same as in Dart, JS, Lua, PHP and Python

4- To delete an element, a built-in function called delete() is used. As a parameter, the key is passed.

5- Modifying an element is the same as in Dart, JS, Lua, PHP and Python

6- To search for the existence of a key, the value of the given key is checked. If there is no such key, the value should be nil. An appropriate message according to the result is displayed.

7- To search for the existence of a value, a built-in function called key() is called. As a parameter, the value that is wanted to check is passed. This function returns the key with the given value. If there is no such value, it returns nil.

8- An each loop is used that iterates through key-value pairs in hash, calling the foo function in every iteration. foo function simply takes key and value as parameter for each iteration and prints the key-value pair

RUST

Code:

```rust
use std::collections::HashMap;

fn main() {

    // Initializing the map
    let mut my_map = HashMap::new();
    my_map.insert("first", "1");
```

```
    my_map.insert("second", "2");
    my_map.insert("third", "3");
    my_map.insert("fourth", "4");
    my_map.insert("fifth", "5");
    println!( "1-Map has been initialized.");

    // Getting value by key
    println!("2-Getting the value of an element by using key");
    let my_value = my_map["first"];
    print!( "the value for the key \"first\" is: ");
    println!("{:?}", my_value);

    // Adding a new element
    println!("3-Adding a new element");
    my_map.insert("sixth", "6");
    println!("Adding a new element with key \"sixth\" and value 6
(String) has been added");

    // Removing an element
    println!("4-Removing an element");
    let k = "third";
    my_map.remove(&k);
    println!("Element with key \"third\" has been removed.");

    // Modifying an element
    println!("5-Modifying an element");
    let my_f = my_map["fourth"];
    print!("The current value for key \"fourth\" is: ");
    println!("{:?}", my_f);
    *my_map.get_mut("fourth").unwrap() = "10";
    let my_f_new = my_map["fourth"];
    print!("After modify, value for key \"fourth\" is: ");
    println!("{:?}", my_f_new);

    // Search by key
    println!("6-Searching existence of key \"third\"");
    let my_key = "third";
    if my_map.contains_key(&my_key)
    {
        let mv = my_map["third"];
        print!("the key \"third\" is in the map with value: ");
        println!("{:?}", mv);
    }
```

```
    else
    {
        println!("the key \"third\" is not in the map");
    }

    // Search by value
    println!("7-Searching existence of value 1");
    let v: String = "1".to_owned();
    let does_contain = my_map.values().any(|&val| *val == v);
    if does_contain == true
    {
        println!("the value 1 is in the map");
    }
    else
    {
        println!("the value 1 is not in the map");
    }

    // For loop
    println!("8-For loop");
    for (key, value) in &my_map {
        foo(key, value);
    }
}

// my function to print pairs
fn foo(k: &&str, v: &&str)
{
    print!("key: ");
    print!("{:?}", k);
    print!(", value: ");
    println!("{:?}", v);
}
```

Output:

```
1-Map has been initialized.
2-Getting the value of an element by using key
the value for the key "first" is: "1"
3-Adding a new element
```

Adding a new element with key "sixth" and value 6 (String) has been added
4-Removing an element
Element with key "third" has been removed.
5-Modifying an element
The current value for key "fourth" is: "4"
After modify, value for key "fourth" is: "10"
6-Searching existence of key "third"
the key "third" is not in the map
7-Searching existence of value 1
the value 1 is in the map
8-For loop
key: "second", value: "2"
key: "fourth", value: "10"
key: "fifth", value: "5"
key: "first", value: "1"
key: "sixth", value: "6"

Explanation:

1- Associative arrays in Rust are called HashMap. To initialize an associative array, an instance of a HashMap is created and elements are, then, inserted, using a built-in function called insert() that takes the key and the value as parameters.

2- Getting a value given a key is the same as all other programming languages used in this homework.

3- Adding a new element has been done by using the insert function (as in initializing).

4- Removing an element has been done by using a built-in function called remove() that takes the key to be deleted as a parameter.

5- To change the value of an element, two built-in functions have been used. The first built-in function is get_mut that takes the key whose value will be modified as a parameter and it is applied on the HashMap. This returns a reference to the value with that key. Second built-in function is unwrap() that is used to change the value of the element with the key previously passed as a parameter.

6- To search for the existence of a key, a built-in function called contains_key that takes the key as a parameter is used.

7- To search for the existence of a value, a built-in function called values() has been used. If any of the values matches with the value that we are looking for, it will return true.

8- A for loop that iterates through keys and values in my_map has been used. In every iteration, foo function is called. foo function, simply, prints the key-value pair.

Evaluation of Languages:

In my opinion, all programming languages in this assignment have so much in common when it comes to associative arrays. What differs among them is the existence of built-in functions or the syntax. When it comes to writability, PHP is the language that I find easiest and most comfortable to do operations with associative arrays because of the good built-in functions such as `unset()`, `array_key_exists()` and `array_search`. Rust is the worst among these, so many symbols and extra symbols to print or pass something as a parameter. So, I think Rust is bad in terms of both readability and writability. Python is a decent one except the fact that with Python3, operations with collections require `list` keyword so much. This affects writability mostly, but readability as well. Dart, Ruby and PHP require identifier symbols to do operations with variables and I think that is an important issue with writability (yet the comfort of PHP's built-in functions suppresses this minor problem). Lua is a decent language that is adaptable if one is familiar with Java. The keywords and built-in functions (such as print function) are so clean. Therefore, I think Lua is the best language in terms of readability. Finally, the built-in functions of Dart and JavaScript are more complicated than the others'.

Learning Strategy:

Programming languages show different approaches of different people to the same problems amazingly. I think there is a strong connection between the programming language that one person is comfortable with and their personality and problem-solving techniques. When I started doing this homework, what I expected was to turn off my brain and write the same code for seven different languages. However, I encountered different object types and functions. That made me ask myself "Would I build a language like that?" Every time and as I did more research about these objects and functions, I have seen the reasons for their existence and their utilization. I avoided using the same syntax. If a language has a built-in function, I tend to use the function to show the differences among the languages. I did not do the homework language by language, I did it operation by operation for every language. I used online compilers or interpreters. The URLs for these compilers and interpreters are as follows,

Dart: https://www.tutorialspoint.com/execute_dart_online.php
JavaScript: https://playcode.io/online-javascript-editor
Lua: https://www.tutorialspoint.com/execute_lua_online.php
PHP: https://www.w3schools.com/php/phptryit.asp?filename=tryphp_compiler
Python: https://www.programiz.com/python-programming/online-compiler/
Ruby: https://www.tutorialspoint.com/execute_ruby_online.php
Rust: https://play.rust-lang.org/