BILKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CS315

Homework Assignment 1

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

Bulut Gözübüyük 21702771

1.0 Dart

1.1 Initialize

Associative arrays in Dart programming language are called Maps. It is a collection of key/value pairs. Using map literals, a map can be declared as follows. Key/value pairs should be in between curly brackets.

1.1.1 Code

```
void main() {
  var credits = {
    'CS319': 4,
    'CS315': 3,
    'MATH225': 4,
    'ENG401': 2
    };
  print(credits);
}
```

1.1.1 Output

```
{CS319: 4, CS315: 3, MATH225: 4, ENG401: 2}
```

1.2 Get the value for a given key

Using the [] operator, value for a given key can be returned when a key is written between the brackets.

1.2.1 Code

```
void main() {
  var credits = {
    'CS319': 4,
    'CS315': 3,
    'MATH225': 4,
    'ENG401': 2
    };
  print("Value for CS315 is: " + credits['CS315'].toString());
}
```

1.2.2 Output

```
Value for CS315 is: 3
```

1.3 Add a new element

By using [] and = operators it associates the new element with the key written between the brackets.

1.3.1 Code

```
void main() {
  var credits = {
    'CS319': 4,
    'CS315': 3,
    'MATH225': 4,
    'ENG401': 2
    };

  credits['GRA131'] = 3;
  print(credits);
}
```

1.3.2 Output

```
{CS319: 4, CS315: 3, MATH225: 4, ENG401: 2, GRA131: 3}
```

1.4 Remove an element

If a key is present in the map, the key/value pair can be deleted using remove() function. The key value should be given inside as a parameter.

1.4.1 Code

```
void main() {
  var credits = {
    'CS319': 4,
    'CS315': 3,
    'MATH225': 4,
    'ENG401': 2
    };

  credits.remove('ENG401');
  print(credits);
}
```

1.4.2 Output

```
{CS319: 4, CS315: 3, MATH225: 4}
```

1.5 Modify the value of an existing element

As in the adding a new element section, a value of a key can be modified in same way using [] and = operators

1.5.1 Code

```
void main() {
  var credits = {
    'CS319': 4,
    'CS315': 3,
    'MATH225': 4,
    'ENG401': 2
    };

  credits['CS319'] = 0;
  print(credits);
}
```

1.5.2 Output

```
{CS319: 0, CS315: 3, MATH225: 4, ENG401: 2}
```

1.6 Search for the existence of a key

To gather a boolean of whether a key exists in a map or not, containsKey() method can be used. The key should be given as an argument.

1.6.1 Code

```
void main() {
  var credits = {
    'CS319': 4,
    'CS315': 3,
    'MATH225': 4,
    'ENG401': 2
    };

  print("CS315 exists: " + credits.containsKey('CS315').toString());
  print("CS101 exists: " + credits.containsKey('CS101').toString());
}
```

1.6.2 Output

```
CS315 exists: true
CS101 exists: false
```

1.7 Search for the existence of a value

To gather a boolean of whether a value exists in a map or not, containsValue() method can be used. The value should be given as an argument.

1.7.1 Code

```
void main() {
  var credits = {
    'CS319': 4,
    'CS315': 3,
    'MATH225': 4,
    'ENG401': 2
    };

  print("5 exists: " + credits.containsValue(5).toString());
  print("4 exists: " + credits.containsValue(4).toString());
}
```

1.7.2 Output

```
5 exists: false
4 exists: true
```

1.8 Loop through an associative array, apply a function, called foo, which simply prints the key-value pair

By using the forEach() method, the “foo” function can be called for each key and value of the associative array.

1.8.1 Code

```
void foo(key, value) {
  print('${key}: ${value}');
}

void main() {
  var credits = {
    'CS319': 4,
    'CS315': 3,
    'MATH225': 4,
    'ENG401': 2
    };
  credits.forEach((key,value) => foo(key,value));
}
```

1.8.2 Output

```
CS319: 4
CS315: 3
MATH225: 4
ENG401: 2
```

1.9 Evaluation of Dart programming language

1.9.1 Readability

Dart provides a well readability because it includes curly braces ({}) to determine scopes. Furthermore, one can read Dart easily because of the large amount of english identifiers.

1.9.2 Writability

While programming associative arrays using Dart programming language, I have noticed that it mostly uses [] operators and english keywords to manage them. Thus, this makes Dart a language that smooths the way of programming with the associative arrays.

2.0 Javascript

2.1 Initialize

Associative arrays in the Javascript programming language are called Objects. It is an array that has keys associated with its values . An object in Javascript can be initialized using {} as follows.

2.1.1 Code

```
var credits = {
   CS319: 4,
   CS315: 3,
   MATH225: 4,
   ENG401: 2
};
console.log(credits);
```

2.1.2 Output

```
{ CS319: 4, CS315: 3, MATH225: 4, ENG401: 2 }
```

2.2 Get the value for a given key

Using the [] operator, value for a given key can be returned when a key is written between the brackets.

2.2.1 Code

```
var credits = {
   CS319: 4,
   CS315: 3,
   MATH225: 4,
   ENG401: 2
};

console.log("Value for CS315 is: " + credits["CS315"]);
```

2.2.2 Output

```
Value for CS315 is: 3
```

2.3 Add a new element

By using [] and = operators it associates the new element with the key written between the brackets. In addition, a new element can be initialized using a dot after object reference and a key.

2.3.1 Code

```
var credits = {
   CS319: 4,
   CS315: 3,
   MATH225: 4,
   ENG401: 2
};

//credits.GRA131 = 3;
credits['GRA131'] = 3;
console.log(credits);
```

2.3.2 Output

```
{ CS319: 4, CS315: 3, MATH225: 4, ENG401: 2, GRA131: 3 }
```

2.4 Remove an element

Deleting a key/value pair of an object can be easily done using the “delete” operator as it can be seen in the following code.

2.4.1 Code

```
var credits = {
   CS319: 4,
   CS315: 3,
   MATH225: 4,
   ENG401: 2
};

// Method 1
delete credits["CS319"];
console.log(credits);

// Method 2
delete credits.CS315;
console.log(credits);
```

2.4.2 Output

```
{ CS315: 3, MATH225: 4, ENG401: 2 }
{ MATH225: 4, ENG401: 2 }
```

2.5 Modify the value of an existing element

As adding a new property in Javascript, a value of a key can be modified in the same way using [] and = operators or in an alternative way using ".".

2.5.1 Code

```
var credits = {
   CS319: 4,
   CS315: 3,
   MATH225: 4,
   ENG401: 2
};

// Method 1
credits["CS319"] = 0
console.log(credits);

// Method 2
credits.CS319 = 99;
console.log(credits);
```

2.5.2 Output

```
{ CS319: 0, CS315: 3, MATH225: 4, ENG401: 2 }
{ CS319: 99, CS315: 3, MATH225: 4, ENG401: 2 }
```

2.6 Search for the existence of a key

To get a boolean of whether a key exists in a map or not, Object's hasOwnProperty() function can be used. The key should be given as an argument.

2.6.1 Code

```
var credits = {
   CS319: 4,
   CS315: 3,
   MATH225: 4,
   ENG401: 2
};

console.log(credits.hasOwnProperty("CS319"));
console.log(credits.hasOwnProperty("CS101"));
```

2.6.2 Output

```
true
false
```

2.7 Search for the existence of a value

To get a boolean about existence of a value in an object, Object.values(objectName).includes(value) method can be used.

2.7.1 Code

```
var credits = {
   CS319: 4,
   CS315: 3,
   MATH225: 4,
   ENG401: 2
};

console.log(Object.values(credits).includes(9));
console.log(Object.values(credits).includes(3));
```

2.7.2 Output

```
false
true
```

2.8 Loop through an associative array, apply a function, called foo, which simply prints the key-value pair

To print in Javascript, console.log() method can be used. To loop in an object for loop can be used as follows.

2.8.1 Code

```
function foo(key, value) {
    console.log(key + ": " + value);
}
var credits = {
   CS319: 4,
   CS315: 3,
   MATH225: 4,
   ENG401: 2
};
for (var k in credits) {
    foo(k, credits[k]);
}
```

2.8.2 Output

```
CS319: 4
CS315: 3
MATH225: 4
ENG401: 2
```

2.9 Evaluation of Javascript programming language

2.9.1 Readability

As in the Dart language, Javascript includes curly braces ({}) to determine scopes. This makes Javascript easier to read. Moreover, since it is close enough to English language, readability of the Javascript programming language becomes way more better.

2.9.2 Writability

As in the Dart programming language, Javascript uses [] operators and a considerable amount of english identifiers to manage the objects (associative arrays). So, it can be concluded that Javascript programming language makes programmers' job easier with the associative arrays.

3.0 Lua

3.1 Initialize

In the Lua programming language associative arrays are called Tables. A table in Lua programming language can be declared in {} curly braces as can be seen in the following code example.

3.1.1 Code

```
local credits = { ["CS319"] = 4,
   ["CS315"] = 3,
   ["MATH225"] = 4,
   ["ENG401"] = 2
   }

for k, v in pairs(credits) do
   print(k, v)
end
```

3.1.2 Output

```
MATH225 4
ENG401  2
CS319   4
CS315   3
```

3.2 Get the value for a given key

By using the keys inside [] operator, value for that key can be returned from the table in Lua programming language.

3.2.1 Code

```
local credits = { ["CS319"] = 4,
   ["CS315"] = 3,
   ["MATH225"] = 4,
   ["ENG401"] = 2
   }

print("Value for CS315 is: " .. credits["CS315"])
```

3.2.2 Output

```
Value for CS315 is: 3
```

3.3 Add a new element

In the Lua programming language, by using [] and = operators it associates the new element with the key written between the brackets.

3.3.1 Code

```
local credits = { ["CS319"] = 4,
   ["CS315"] = 3,
   ["MATH225"] = 4,
   ["ENG401"] = 2
   }
credits["GRA131"] = 3

for k, v in pairs(credits) do
   print(k, v)
end
```

3.3.2 Output

```
ENG401  2
CS315   3
GRA131  3
CS319   4
MATH225 4
```

3.4 Remove an element

“nil” is used to delete a key/value pair in the Lua programming language. Setting tableName[key] = nil will delete the desired key/value pair from the table.

3.4.1 Code

```
local credits = { ["CS319"] = 4,
   ["CS315"] = 3,
   ["MATH225"] = 4,
   ["ENG401"] = 2
   }
credits["ENG401"] = nil

for k, v in pairs(credits) do
   print(k, v)
end
```

3.4.2 Output

```
MATH225 4
CS319   4
CS315   3
```

3.5 Modify the value of an existing element

In the Lua programming language a value of a key can be modified in the same way using [] and = operators as adding a new element to the table.

3.5.1 Code

```
local credits = { ["CS319"] = 4,
   ["CS315"] = 3,
   ["MATH225"] = 4,
   ["ENG401"] = 2
   }

credits["CS315"] = 0
print("Value for CS315 is: " .. credits["CS315"])
```

3.5.2 Output

```
Value for CS315 is: 0
```

3.6 Search for the existence of a key

In the Lua programming language, to determine whether a key/value pair exists in the table, comparing the value for a key with the “nil” is enough. The comparison will return true if it exists. The comparison can be done using ~= operator.

3.6.1 Code

```
local credits = { ["CS319"] = 4,
   ["CS315"] = 3,
   ["MATH225"] = 4,
   ["ENG401"] = 2
   }

print(credits["CS101"] ~= nil)
print(credits["CS315"] ~= nil)
```

3.6.2 Output

```
false
true
```

3.7 Search for the existence of a value

There is no function that checks whether a value exists in a table. To do that, I have written a simple function that loops through all key value pairs and checks if it matches with the requested value as can be seen below.

3.7.1 Code

```
local function containsValue(table, value)
  for k,v in pairs(table) do
    if v == value then
      return true
    end
  end
  return false
end

local credits = { ["CS319"] = 4,
   ["CS315"] = 3,
   ["MATH225"] = 4,
   ["ENG401"] = 2
   }

--Searching for 9 and 3

print(containsValue(credits, 9))
print(containsValue(credits, 3))
```

3.7.2 Output

```
false
true
```

3.8 Loop through an associative array, apply a function, called foo, which simply prints the key-value pair

In Lua programming language, to loop through a table with key/value pairs “for key, value in pairs(tableName)” can be used as follows. The function is defined using “local function” keywords. In each loop foo function is called with 2 arguments: key and value.

3.8.1 Code

```
local function foo(key, value)
   print(key, value)
end

local credits = { ["CS319"] = 4,
   ["CS315"] = 3,
   ["MATH225"] = 4,
   ["ENG401"] = 2
   }

for k, v in pairs(credits) do
   foo(k,v)
end
```

3.8.2 Output

```
MATH225 4
ENG401  2
CS319   4
CS315   3
```

3.9 Evaluation of Lua programming language

3.9.1 Readability

As in the Dart and in the Javascript programming languages, Lua programming language has curly braces to determine scopes. Compared to Dart and the Javascript languages, Lua provides almost the same readability as the others since Lua also uses a considerable amount of english keywords.

3.9.2 Writability

As in the Dart and in the Javascript programming language, Lua programming language also uses [] operators and a considerable amount of english words to manage the tables. It can be concluded that Lua is showing the same writability as Javascript and Dart because they are showing similar features.

4.0 PHP

4.1 Initialize

In the PHP programming language, an array can be declared in () parentheses and using “=>” operators as can be seen in the following code example.

4.1.1 Code

```
<?php
$credits = array("CS319" => 4,
            "CS315" => 3,
            "MATH225" => 4,
            "ENG401" => 2
            );
foreach( $credits as $k => $v )
   echo $k.": ".$v."\n";
```

4.1.2 Output

```
CS319: 4
CS315: 3
MATH225: 4
ENG401: 2
```

4.2 Get the value for a given key

By using the $ sign and keys inside [] operator, value for that key can be returned from the array in PHP programming language.

4.2.1 Code

```
<?php
$credits = array("CS319" => 4,
            "CS315" => 3,
            "MATH225" => 4,
            "ENG401" => 2
            );
echo "Value for CS315 is ${credits["CS315"]}";
```

4.2.2 Output

```
Value for CS315 is 3
```

4.3 Add a new element

In the PHP programming language, by using [] and = operators it associates the new element with the key written between the brackets. Dollar sign $ should be used before the array name.

4.3.1 Code

```
<?php
$credits = array("CS319" => 4,
            "CS315" => 3,
            "MATH225" => 4,
            "ENG401" => 2
            );
$credits['GRA131'] = 3;

foreach( $credits as $k => $v )
   echo $k.": ".$v."\n";
```

4.3.1 Output

```
CS319: 4
CS315: 3
MATH225: 4
ENG401: 2
GRA131: 3
```

4.4 Remove an element

In the PHP programming language the unset function is to remove elements in the array. A sample usage can be seen in the program below.

4.4.1 Code

```
<?php
$credits = array("CS319" => 4,
            "CS315" => 3,
            "MATH225" => 4,
            "ENG401" => 2
            );
unset($credits["ENG401"]);
foreach( $credits as $k => $v )
   echo $k.": ".$v."\n";
```

4.4.2 Output

```
CS319: 4
CS315: 3
MATH225: 4
```

4.5 Modify the value of an existing element

In the PHP programming language, modifying the value of an existing element can be done using the same way as adding an element by using [] and = operators and the key is written between the brackets. Dollar sign $ should be used before the array name.

4.5.1 Code

```
<?php
$credits = array("CS319" => 4,
            "CS315" => 3,
            "MATH225" => 4,
            "ENG401" => 2
            );
$credits["CS315"] = 999;

foreach( $credits as $k => $v )
   echo $k.": ".$v."\n";
```

4.5.2 Output

```
CS319: 4
CS315: 999
MATH225: 4
ENG401: 2
```

4.6 Search for the existence of a key

To determine whether a key exists or not in the PHP programming language, array_key_exists(keyName, arrayName) method can be used. Sample usage can be seen below. It prints true if it exists and it prints false if it does not exist.

4.6.1 Code

```
<?php
$credits = array("CS319" => 4,
            "CS315" => 3,
            "MATH225" => 4,
            "ENG401" => 2
            );
echo array_key_exists("CS101", $credits) ? "true\n": "false\n";
echo array_key_exists("CS315", $credits) ? "true\n" : "false\n";
```

4.6.2 Output

```
false
true
```

4.7 Search for the existence of a value

In the PHP programming language to determine whether a value is in array or not, in_array(value, arrayName) method can be used. The true false message can be printed the same way as before.

4.7.1 Code

```
<?php
$credits = array("CS319" => 4,
            "CS315" => 3,
            "MATH225" => 4,
            "ENG401" => 2
            );

echo in_array(9, $credits) ? "true\n": "false\n";
echo in_array(3, $credits) ? "true\n": "false\n";
```

4.7.2 Output

```
false
true
```

4.8 Loop through an associative array, apply a function, called foo, which simply prints the key-value pair

In the PHP programming language, to loop through a table with key/value pairs “foreach” can be used as follows. In each loop foo function is called with 2 arguments: key and value.

4.8.1 Code

```
<?php
function foo($key, $value) {
   echo $key.": ".$value."\n";
}
$credits = array("CS319" => 4,
            "CS315" => 3,
            "MATH225" => 4,
            "ENG401" => 2
            );

foreach( $credits as $k => $v )
   foo($k,$v);
```

4.8.2 Output

```
CS319: 4
CS315: 3
MATH225: 4
ENG401: 2
```

4.9 Evaluation of PHP programming language

4.9.1 Readability

Because PHP needs a dollar sign before variables, this makes PHP programming language harder to read.

4.9.2 Writability

Because PHP needs a dollar sign before variables, this makes PHP programming language harder to write. Furthermore it needs a php tag at the start of the program. In addition, there is no true or false so that an extra programming is needed to print boolean expressions.

5.0 Python3

5.1 Initialize

Associative arrays in the Python programming language are called Dictionaries. It is an array that has keys associated with its values . A dictionary in Python programming language can be initialized using {} as follows.

5.1.1 Code

```
credits = {
  "CS319": 4,
  "CS315": 3,
  "MATH225": 4,
  "ENG401": 2
}

print(credits)
```

5.1.2 Output

```
{'ENG401': 2, 'CS315': 3, 'CS319': 4, 'MATH225': 4}
```

5.2 Get the value for a given key

Using the [] operator, value for a given key can be returned when a key is written between the brackets.

5.2.1 Code

```
credits = {
  "CS319": 4,
  "CS315": 3,
  "MATH225": 4,
  "ENG401": 2
}

print("Value for CS315 is", credits["CS315"])
```

5.2.2 Output

```
Value for CS315 is 3
```

5.3 Add a new element

By using [] and = operators it associates the new element with the key written between the brackets.

5.3.1 Code

```
credits = {
  "CS319": 4,
  "CS315": 3,
  "MATH225": 4,
  "ENG401": 2
}

credits["GRA131"] = 3

print(credits)
```

5.3.2 Output

```
{'CS315': 3, 'ENG401': 2, 'CS319': 4, 'MATH225': 4, 'GRA131': 3}
```

5.4 Remove an element

The key/value pair can be deleted using the “del” keyword. The key value should be given inside as a parameter.

5.4.1 Code

```
credits = {
  "CS319": 4,
  "CS315": 3,
  "MATH225": 4,
  "ENG401": 2
}

del credits["CS315"]

print(credits)
```

5.4.2 Output

```
{'CS319': 4, 'MATH225': 4, 'ENG401': 2}
```

5.5 Modify the value of an existing element

As in the adding a new element section, a value of a key can be modified in same way using [] and = operators

5.5.1 Code

```
credits = {
  "CS319": 4,
  "CS315": 3,
  "MATH225": 4,
  "ENG401": 2
}

credits["CS315"] = 999

print(credits)
```

5.5.2 Output

```
{'MATH225': 4, 'CS315': 999, 'CS319': 4, 'ENG401': 2}
```

5.6 Search for the existence of a key

To gather a boolean about existence of a key in Python programming language, “in” keyword can be used. Sample usage can be seen below. (key in dictName)

```
credits = {
  "CS319": 4,
  "CS315": 3,
  "MATH225": 4,
  "ENG401": 2
}

print("CS101" in credits)
print("CS315" in credits)
```

5.6.2 Output

```
False
True
```

5.7 Search for the existence of a value

Dictionary.values() returns a list of values in that dictionary. Using the “in” keyword again we can get a boolean whether a value exists in that dictionary or not.

5.7.1 Code

```
credits = {
  "CS319": 4,
  "CS315": 3,
  "MATH225": 4,
  "ENG401": 2
}

print(0 in credits.values())
print(3 in credits.values())
```

5.7.2 Output

False
True

5.8 Loop through an associative array, apply a function, called foo, which simply prints the key-value pair

Using the “def” keyword, a python function can be defined. And its arguments are written in parentheses. Furthermore, by using “for i in dictName” we can loop through in a dictionary.

5.8.1 Code

```
def foo(key, value):
    print(key, value)

credits = {
  "CS319": 4,
  "CS315": 3,
  "MATH225": 4,
  "ENG401": 2
}

for key in credits:
    foo(key, credits[key])
```

5.8.2 Output

CS315 3
ENG401 2
MATH225 4
CS319 4

5.9 Evaluation of Python programming language

5.9.1 Readability

Up to this point in time, it can be claimed that Python is the easiest programming language in the context of Readability since it is too close to the English language.

5.9.2 Writability

Python needs less symbols since it uses spaces instead of curly braces for indentation. Furthermore, assignment or retrieval of dictionary values are so simple and it is too close to the english language. Thus, it can be concluded that python programming language is successful in the context of writability.

6.0 Ruby

6.1 Initialize

Associative arrays in Ruby programming language are called Hash. It is a collection of key/value pairs. Key/value pairs should be in between curly brackets.

6.1.1 Code

```
credits = {"CS319" => 4,
   "CS315" => 3,
   "MATH225" => 4,
   "ENG401" => 2
}
puts credits
```

6.1.2 Output

```
{"CS319"=>4, "CS315"=>3, "MATH225"=>4, "ENG401"=>2}
```

6.2 Get the value for a given key

Using the [] operator, value for a given key can be returned when a key is written between the brackets.

6.2.1 Code

```
credits = {"CS319" => 4,
   "CS315" => 3,
   "MATH225" => 4,
   "ENG401" => 2
}
puts "Value for CS315 is #{credits["CS315"]}"
```

6.2.2 Output

```
Value for CS315 is 3
```

6.3 Add a new element

By using [] and = operators it associates the new element with the key written between the brackets.

6.3.1 Code

```
credits = {"CS319" => 4,
   "CS315" => 3,
   "MATH225" => 4,
   "ENG401" => 2
}

credits["GRA131"] = 3

puts credits
```

6.3.2 Output

```
{"CS319"=>4, "CS315"=>3, "MATH225"=>4, "ENG401"=>2, "GRA131"=>3}
```

6.4 Remove an element

In the Ruby programming language, the key/value pair can be deleted using delete() function. The key value should be given inside as a parameter.

6.4.1 Code

```
credits = {"CS319" => 4,
   "CS315" => 3,
   "MATH225" => 4,
   "ENG401" => 2
}

credits.delete("CS315")

puts credits
```

6.4.2 Output

```
{"CS319"=>4, "MATH225"=>4, "ENG401"=>2}
```

6.5 Modify the value of an existing element

As adding a new element to the hash, a value of a key can be modified in same way using [] and = operators

6.5.1 Code

```
credits = {"CS319" => 4,
   "CS315" => 3,
   "MATH225" => 4,
   "ENG401" => 2
}

credits["CS315"] = 999

puts credits
```

6.5.2 Output

```
{"CS319"=>4, "CS315"=>999, "MATH225"=>4, "ENG401"=>2}
```

6.6 Search for the existence of a key

To gather a boolean of whether a key exists in a map or not, key?() method can be used. The key should be given as an argument.

6.6.1 Code

```
credits = {"CS319" => 4,
   "CS315" => 3,
   "MATH225" => 4,
   "ENG401" => 2
}

puts credits.key?("CS101")
puts credits.key?("CS315")
```

6.6.2 Output

```
false
true
```

6.7 Search for the existence of a value

To gather a boolean of whether a value exists in a map or not, has_value?() method can be used. The value should be given as an argument.

6.7.1 Code

```
credits = {"CS319" => 4,
   "CS315" => 3,
   "MATH225" => 4,
   "ENG401" => 2
}

puts credits.has_value?(0)
puts credits.has_value?(3)
```

6.7.2 Output

```
false
true
```

6.8 Loop through an associative array, apply a function, called foo, which simply prints the key-value pair

By using the hashName.each and “do”, the “foo” function can be called for each key and value of the associative array.

6.8.1 Code

```
def foo(key, value)
  puts "#{key}: #{value}"
end

credits = {"CS319" => 4,
   "CS315" => 3,
   "MATH225" => 4,
   "ENG401" => 2
}

credits.each do |k,v|
   foo(k,v)
end
```

6.8.2 Output

```
CS319: 4
CS315: 3
MATH225: 4
ENG401: 2
```

6.9 Evaluation of Ruby programming language

6.9.1 Readability

Since Ruby programming language needs # operator, this makes language harder to read. Furthermore, this also applies to pipe usage in for loops.

6.9.2 Writability

As in most of the languages I have analyzed, Ruby seems to have similar writability since Ruby programming language uses [] operators and english keywords to manage hashes(associative arrays).

7.0 Rust

7.1 Initialize

Associative arrays in Rust programming language are called Maps. It is a collection of key/value pairs. Key/value pairs should be in between curly brackets. To create a new hashmap following code sample can be used. If a mutable map is needed, then a “mut” keyword should be written after let.

7.1.1 Code

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

fn main() {
   let credits: HashMap<&str, i32> =
   [("CS319", 4),
   ("CS315", 3),
   ("MATH225", 4),
   ("ENG401", 2)].iter().cloned().collect();

   println!("{:?}", credits);
}
```

7.1.2 Output

```
{"CS315": 3, "CS319": 4, "MATH225": 4, "ENG401": 2}
```

7.2 Get the value for a given key

In the Rust programming language, a value can be accessed using HashMap.get() method.

7.2.1 Code

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

fn main() {
   let credits: HashMap<&str, i32> =
   [("CS319", 4),
   ("CS315", 3),
   ("MATH225", 4),
   ("ENG401", 2)].iter().cloned().collect();

   println!("Value for CS315 is {:?}", credits.get("CS315").unwrap());
}
```

7.2.2 Output

```
Value for CS315 is 3
```

7.3 Add a new element

To add a new key/value pair to HashMap in Rust programming language, HashMap.insert(key,value) method can be used.

7.3.1 Code

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

fn main() {
   let mut credits: HashMap<&str, i32> =
   [("CS319", 4),
   ("CS315", 3),
   ("MATH225", 4),
   ("ENG401", 2)].iter().cloned().collect();

   credits.insert("GRA131",3);

   println!("{:?}", credits);
}
```

7.3.2 Output

```
{"CS319": 4, "ENG401": 2, "CS315": 3, "MATH225": 4, "GRA131": 3}
```

7.4 Remove an element

To remove an element from HashMap, remove() function can be used.

7.4.1 Code

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

fn main() {
   let mut credits: HashMap<&str, i32> =
   [("CS319", 4),
   ("CS315", 3),
   ("MATH225", 4),
   ("ENG401", 2)].iter().cloned().collect();

   credits.remove("CS315");

   println!("{:?}", credits);
}
```

7.4.2 Output

```
{"ENG401": 2, "CS319": 4, "MATH225": 4}
```

7.5 Modify the value of an existing element

To modify value of an existing element in Rust, "*mapName.get_mut(key)" can be used or the key value pair can be reinserted as in add a new element section. If reinserted, the value will be overwritten.

7.5.1 Code

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

fn main() {
    let mut credits: HashMap<&str, i32> =
    [("CS319", 4),
    ("CS315", 3),
    ("MATH225", 4),
    ("ENG401", 2)].iter().cloned().collect();

    *credits.get_mut("CS315").unwrap() = 999;

    println!("{:?}", credits);
}
```

7.5.2 Output

```
{"MATH225": 4, "ENG401": 2, "CS319": 4, "CS315": 999}
```

7.6 Search for the existence of a key

To search for the existence of a key, HashMap.contains_key() method can be used. The key should be written as an argument.

7.6.1 Code

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

fn main() {
    let credits: HashMap<&str, i32> =
    [("CS319", 4),
    ("CS315", 3),
    ("MATH225", 4),
    ("ENG401", 2)].iter().cloned().collect();

    println!("{:?}", credits.contains_key("CS101"));
    println!("{:?}", credits.contains_key("CS315"));
}
```

7.6.2 Output

```
false
true
```

7.7 Search for the existence of a value

In the Rust programming language, the existence of a value in a HashMap can be check using “HashMap.values.any(|&v| v == valueToBeSearched))”.

7.7.1 Code

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

fn main() {
   let credits: HashMap<&str, i32> =
   [("CS319", 4),
   ("CS315", 3),
   ("MATH225", 4),
   ("ENG401", 2)].iter().cloned().collect();

   println!("{:?}", credits.values().any(|&value| value == 9));
   println!("{:?}", credits.values().any(|&value| value == 3 ));
}
```

7.7.2 Output

false
true

7.8 Loop through an associative array, apply a function, called foo, which simply prints the key-value pair

By using the for and in, the “foo” function can be called for each key and value of the associative array.

7.8.1 Code

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

fn foo(key:&str,value:&i32){
   println!("{}: {}", key, value);
}
fn main() {
   let credits: HashMap<&str, i32> =
   [("CS319", 4),
   ("CS315", 3),
   ("MATH225", 4),
   ("ENG401", 2)].iter().cloned().collect();

   for (k, v) in &credits {
      foo(k,v);
   }
}
```

7.8.2 Code

CS319: 4
MATH225: 4
CS315: 3
ENG401: 2

7.9 Evaluation of Rust programming language

7.9.1 Readability

Due to the fact that Rust programming language has an excessive amount of extra symbols and functions needed to execute certain operations compared to other languages, it can be claimed that Rust is the hardest programming language in terms of readability.

7.9.2 Writability

As mentioned in the readability section, Rust has an excessive amount of symbols and functions for even simple operations like printing to console etc. This makes Rust programming language harder to write.

8.0 Learning Strategy

While learning all of 7 languages, I aimed to adhere to the official documentations of the programming languages. If I have issues after looking at the official documentation, I search in stack overflow then I find solutions to my problems. Those 2 approaches lead me to complete this homework so it can be claimed that they are enough.

URLs of the official documentations

1. Dart https://api.dart.dev/stable/2.10.4/index.html
2. Javascript by Mozilla https://developer.mozilla.org/en-US/docs/Web/JavaScript
3. Lua https://www.lua.org/
4. PHP https://www.php.net/docs.php
5. Python3 https://docs.python.org/3/
6. Ruby https://www.ruby-lang.org/en/documentation/
7. Rust https://doc.rust-lang.org/

URLs of the online compiler/interpreters

8. Dart https://dartpad.dev/
9. Javascript https://repl.it/languages/javascript
10. Lua https://repl.it/languages/lua
11. PHP https://sandbox.onlinephpfunctions.com/
12. Python3 https://www.onlinegdb.com/online_python_compiler
13. Ruby https://repl.it/languages/ruby
14. Rust https://repl.it/languages/rust

9.0 Conclusion

In my opinion, in terms of writability and readability, the best language for me is Python and the worst is Ruby. Because Python is too close to english language and Ruby has an excessive amount of symbols and functions required to do simple operations on associative arrays.