CS 315 HOMEWORK 1

ALP TUĞRUL AĞÇALI

21801799

SECTION 3

1 Dart

Sample Code

```
void foo (var input1, var input2){
	print(input1 + " " + input2);
}
void main() {
	var workedTime = {'Alp' : '8 hours', 'Tuna' : 'hours'};
	print(workedTime);
	print(workedTime['Alp']);
	workedTime['Emrehan'] = '8 hours';
	print(workedTime);
	workedTime.remove('Alp');
	print(workedTime);
	workedTime['Tuna'] = '8 hours';
	print(workedTime);
	print(workedTime.containsKey('Alp'));
	print(workedTime.containsKey('Tuna'));
	print(workedTime.containsValue('7 hours'));
	print(workedTime.containsValue('8 hours'));
	for(var temp in workedTime.entries){
		foo(temp.key,temp.value);
	}
}
```

Output

```
{Alp: 8 hours, Tuna: hours}
8 hours
{Alp: 8 hours, Tuna: hours, Emrehan: 8 hours}
{Tuna: hours, Emrehan: 8 hours}
{Tuna: 8 hours, Emrehan: 8 hours}
false
true
false
true
Tuna 8 hours
Emrehan 8 hours
```

In my example of Dart language, I firstly initialize an associative array (map) which holds names of the students as key and the time they worked as value. Then, I get the value of first student. I added new student, remove one student by using remove function and modify one of the student whose value is written wrongly by me at the beginning. Then I checked existence of two key, one of them did not exist and other one did. I use containsKey function to check. After that I checked the existence of an value in the associative array, again one of them existed and other did not, using the containsValue function. Lastly, I wrote a function called foo which simply prints the key and value of associative array, I use for loop to put every key and value in this function. In the for loop I could use the Dart's property to get every entry (key and value).

2. JavaScript

Sample Code

```
var workedTime = {"Alp": "8 hours", "Tuna": "hours"};

console.log(workedTime);

console.log(workedTime.Alp);

workedTime.Emrehan = "8 hours";

console.log(workedTime);

delete workedTime["Alp"]

console.log(workedTime);

workedTime.Tuna = "8 hours";

console.log(workedTime);

console.log("Alp" in workedTime);

console.log("Tuna" in workedTime);

function containsValue(array, value) {
	for (var key in array) {
		if(array[key] === value)
			return true;
	}
```

```
        return false;
}
console.log(containsValue(workedTime, "hours"));
console.log(containsValue(workedTime, "8 hours"));
function foo(key, value){
console.log(key + " " + value)
}
for(var key in workedTime){
        foo(key, workedTime[key])
}
```

Output

```
{ Alp: '8 hours', Tuna: 'hours' }
8 hours
{ Alp: '8 hours', Tuna: 'hours', Emrehan: '8 hours' }
{ Tuna: 'hours', Emrehan: '8 hours' }
{ Tuna: '8 hours', Emrehan: '8 hours' }
false
true
false
true
Tuna 8 hours
Emrehan 8 hours
```

In my example of JavaScript language, I firstly initialize an associative array which holds names of the students as key and the time they worked as value again. Then, I get the value of first student. I added new student, remove one student by using delete operator and modify one of the student whose value is written wrongly by me at the beginning. Then I checked existence of two key, one of them did not exist and other one did. I use JavaScript's property to check keys of an associative array. (key in assArray) I define a new function in order to check values in associative array. Lastly, I wrote a function called foo which simply prints the key and value of associative array, I use for loop to put every key and value in this function. I could use approximately same code with dart in order to get every entry of the associative array in for loop. In this time I did not get entries directly but get keys in order.

3. Lua

Sample Code

```
workedTime = {["Alp"] = "8 hours", ["Tuna"] = "hours"}
for i,v in pairs(workedTime) do print(i,v) end
print(workedTime["Alp"])
workedTime["Emrehan"] = "8 hours"
for i,v in pairs(workedTime) do print(i,v) end
workedTime["Alp"] = nil
for i,v in pairs(workedTime) do print(i,v) end
workedTime["Tuna"] = "8 hours"
for i,v in pairs(workedTime) do print(i,v) end
print(workedTime["Alp"] ~= nil)
print(workedTime["Tuna"] ~= nil)
function table_contains(tbl, x)
   found = false
   for _, v in pairs(tbl) do
      if v == x then
         found = true
      end
   end
   return found
end
print(table_contains(workedTime, "8 hours"))
print(table_contains(workedTime, "hours"))
function foo(key, value)
   print(key, value)
end
for i,v in pairs(workedTime) do foo(i,v) end
```

Output

```
Alp	8 hours
Tuna	hours
8 hours
Alp	8 hours
```

Tuna hours

Emrehan 8 hours

Tuna hours

Emrehan 8 hours

Tuna 8 hours

Emrehan 8 hours

false

true

true

false

Tuna 8 hours

Emrehan 8 hours

In my example of Lua language, I firstly initialize an associative array (table) which holds names of the students as key and the time they worked as value again. Then, I get the value of first student. I added new student, remove one student by saying it's value is equal to nil and modify one of the student whose value is written wrongly by me at the beginning. Then I checked existence of two key, one of them did not exist and other one did. I use Lua's property to check keys of an associative array. (key~=nil) I define a new function in order to check values in associative array. Lastly, I wrote a function called foo which simply prints the key and value of associative array, I use for loop to put every key and value in this function. I use for loop property of lua in order to print the elements and call foo function.

4. PHP

Example Code

```
<?php

$workedHour = array("Alp"=>"8 Hours", "Tuna"=>"Hours");

print_r($workedHour);

echo $workedHour["Alp"], "\n";

$workedHour["Emrehan"] = "8 Hours";

print_r($workedHour);

unset($workedHour['Alp']);

print_r($workedHour);

$workedHour["Tuna"] = "8 Hours";

print_r($workedHour);

if (array_key_exists("Alp", $workedHour)){
```

```
   echo "true", "\n";
}
else {
   echo "false", "\n";
}
if (array_key_exists("Tuna", $workedHour)){
   echo "true", "\n";
}
else {
   echo "false", "\n";
}
if (in_array("hours", $workedHour)){
   echo "true", "\n";
}
else {
   echo "false", "\n";
}
if (in_array("8 Hours", $workedHour)){
   echo "true", "\n";
}
else {
   echo "false", "\n";
}
function foo($key, $value) {
  echo $key, " ", $value, "\n";
}
foreach ($workedHour as $person => $time){
   foo($person, $time);
}
?>
```

Output

```
Array
(
    [Alp] => 8 Hours
    [Tuna] => Hours
)
8 Hours
Array
(
    [Alp] => 8 Hours
    [Tuna] => Hours
    [Emrehan] => 8 Hours
)
Array
(
    [Tuna] => Hours
    [Emrehan] => 8 Hours
)
Array
(
    [Tuna] => 8 Hours
    [Emrehan] => 8 Hours
)
false
true
false
true
Tuna 8 Hours
Emrehan 8 Hours
```

In my example of PHP language, I firstly initialize an associative array which holds names of the students as key and the time they worked as value again. Then, I get the value of first student. I added new student, remove one student using unset function and modify one of the student whose value is written wrongly by me at the beginning. Then I checked existence of two key, one of them did not exist and other one did. I use Php's property to check keys of an associative array however I could not use it while printing the result, so I use echo to print "true" and "false". Again I use Php's in_array function to check if a value exist in associative array and use echo to print results. Lastly, I wrote a function called foo which simply prints the key and value of associative array, I use foreach loop to put every key and value in this function.

5. Python

Example Code

```
workedTime = {'Alp' : '8 hours', 'Tuna' : 'hours'}

print(workedTime)

print(workedTime['Alp'])

workedTime['Emrehan'] = '8 hours'

print(workedTime)

workedTime.pop('Alp')

print(workedTime)
```

```
workedTime.update({'Tuna' : '8 hours'})

print(workedTime)

print('Alp' in workedTime)

print('Tuna' in workedTime)

print('hours' in workedTime.values())

print('8 hours' in workedTime.values())

def foo(input1, input2):

   print(input1 + " " + input2)

for key, value in workedTime.items():

   foo(key, value)
```

Output

```
{'Alp': '8 hours', 'Tuna': 'hours'}

8 hours

{'Alp': '8 hours', 'Tuna': 'hours', 'Emrehan': '8 hours'}

{'Tuna': 'hours', 'Emrehan': '8 hours'}

{'Tuna': '8 hours', 'Emrehan': '8 hours'}

False

True

False

True

Tuna 8 hours

Emrehan 8 hours
```

In my example of Phyton language, I firstly initialize an associative array which holds names of the students as key and the time they worked as value again. Then, I get the value of first student. I added new student, remove one student using pop function and modify one of the student whose value is written wrongly by me by using update function at the beginning. Then I checked existence of two key, one of them did not exist and other one did. I use phyton's operator in to check keys of an associative array. Same with the keys I use in operator to check wheter values exist or not, however, this time I use values function to get associative array's values. Lastly I define a function called foo that prints key and value. In order to get keyy and value I use for loop.

6. Ruby

Example Code

```
workedTime = {'Alp' => '8 hours', 'Tuna' => 'Hours'}

puts workedTime
```

```
puts workedTime['Alp']

workedTime['Emrehan'] = '8 hours'

puts workedTime

workedTime.delete('Alp')

puts workedTime

workedTime['Tuna'] = '8 hours'

puts workedTime

puts workedTime.key?('Alp')

puts workedTime.key?('Emrehan')

puts workedTime.value?('Hours')

puts workedTime.value?('8 hours')

def foo(key, value)

   puts key + ' ' + value

end

workedTime.each { |key, value| foo(key,value) }
```

Output

```
{"Alp"=>"8 hours", "Tuna"=>"Hours"}

8 hours

{"Alp"=>"8 hours", "Tuna"=>"Hours", "Emrehan"=>"8 hours"}

{"Tuna"=>"Hours", "Emrehan"=>"8 hours"}

{"Tuna"=>"8 hours", "Emrehan"=>"8 hours"}

false

true

false

true

Tuna 8 hours

Emrehan 8 hours
```

In my example of Ruby language, I firstly initialize an associative array which holds names of the students as key and the time they worked as value again. Then, I get the value of first student. I added new student, remove one student using delete function and modify one of the student whose value is written wrongly by me at the beginning. Then I checked existence of two key, one of them did not exist and other one did. I use Ruby's function key? to check keys of an associative array. Same

with the keys I use value? Function to check wheter values exist or not. Lastly I define a function called foo that prints key and value. In order to get keyy and value I use each loop.

7. Rust

Example Code

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

fn foo(key: &str, val: &str){
   println!("{} {}", key, val);
}

fn main() {
   let mut workedTime=HashMap::new();
   workedTime.insert("Alp", "8 hours");
   workedTime.insert("Tuna", "hours");
   println!("{:?}",workedTime);
   println!("{:?}", workedTime.get(&"Alp"));
   workedTime.insert("Emrehan", "8 hours");
   println!("{:?}",workedTime);
   workedTime.remove(& "Alp");
   println!("{:?}",workedTime);
   workedTime.insert("Tuna","8 hours");
   println!("{:?}",workedTime);
   println!("{:?}", workedTime.contains_key( & "Alp"));
   println!("{:?}", workedTime.contains_key( & "Tuna"));
   println!("{:?}", workedTime.values().any(|&val| val == "8 hours"));
   println!("{:?}", workedTime.values().any(|&val| val == "hours"));
   for (key, val) in workedTime.iter() {
      foo(key,val);
   }
}
```

Output

```
{"Alp": "8 hours", "Tuna": "hours"}
Some("8 hours")
{"Alp": "8 hours", "Emrehan": "8 hours", "Tuna": "hours"}
```

```
{"Emrehan": "8 hours", "Tuna": "hours"}
{"Emrehan": "8 hours", "Tuna": "8 hours"}
false
true
true
false
Emrehan 8 hours
Tuna 8 hours
```

In my example of Rust language, I firstly initialize an associative array which holds names of the students as key and the time they worked as value and I put first two values with insert function. Then, I get the value of first student. I added new student again with insert function, remove one student using remove function and modify one of the student, whose value is written wrongly by me, by using insert function again at the beginning. Then I checked existence of two key, one of them did not exist and other one did. I use Rust' function contains_key to check keys of an associative array. To check values of the associative array I firstly use values function to et values and any function to check whether values exist in the associative array. Lastly I define a function called foo that prints key and value. In order to get key and value I use for loop.

Evaluation

1. **Dart**

In my opinion Dart is good enough for using associative arrays. Because, firstly, it allows to use functions to make a loop or check key and value. In addition it is easy to write and read because it has simple structure. I think the best way of Dart is it allows to get entries, key and value together.

2. **JavaScript**

In my opinion Javascript is not one of the best languages for associative arrays. Becaues, altough it has a functions for for loops and checking keys it does not have a function to check values. Another negative property of the Javascript is == and === may be confusing. In addition to this, It accepts both quotes and apostrophe for strings and this affect the simplicity.

3. **Lua**

In my opinion Lua is not useful language for associative arrays because of the insufficient functions like javascript. In addition making nil while deleting and using ~= while checking existence are makes readabilty and writability poor.

4. **PHP**

I think PHP is making good job with the associative arrays. However, it does not print true and false to console. Using $ sign at the beginning of the variables is make writability hard. However, foreach loop is very good in readability and useful because we can assign variables to key and value directly.

5. **Phyton**

In my opinion Phyton is making very good job about associative arrays too. It is easy to write thanks to functions and operators. In addition to this update function makes updating clear, easy to read. It accepts both quotes and apostrophe for strings and this is negative aspect for simplicity.

6. **Ruby**

Ruby is one of the best languages for associative arrays. It is very simple with functions. It's functions delete, key?, value? And rach loop are easy to write and read. Functions start with def and ends with end this makes readability and writability good. Again it accepts both quotes and apostrophe for strings and this is negative aspect for simplicity.

7. **Rust**

I think Rust is worst language for associative arrays. Because it is hard to write and read. Altough it has enough functions, writing and reading them is hard. Like C++ user had to use & operators. Printing becomes hard when user try to use variables here.,

To sum up, according to reasons that I said above I think best language that I experienced in this homework was Ruby.

Learning Strategies

In order to do this homework I studied the languages on www.geeksforgeeks.org and www.w3schools.com. I searched tasks on stackoverflow.com when I can't find them in geeksforgeeks and w3schools. Additionally, I use https://www.tutorialspoint.com/ to find informations about these languages. The online compiler/interpreters I used are:

- https://dartpad.dev/?id
- https://www.programiz.com/javascript/online-compiler/
- https://www.tutorialspoint.com/execute_lua_online.php
- https://paiza.io/projects/Rl3tOaOXXCzP8YaE6SszqA
- https://www.programiz.com/python-programming/online-compiler/
- https://www.tutorialspoint.com/execute_ruby_online.php
- https://play.rust-lang.org/