CS 315 Homework 1

Section: 2
Barkın Saday - 21902967

Note: Codes, outputs and the way our program works in all languages are very similar to each other. Explanation for Python (part 1.3) is given in more detail. Thus, explanation for other languages contains references to Python's explanation segment in order to not repeat same explanations.

1) Python

1.1) Code:

```
#1) Initialize
grades = {'Barkin': 95, 'Ahmet': 65, 'Ayse': 78, 'Dilruba': 100}

#2) Get Value For a Key (2 different ways)
print(grades['Barkin'])
print(grades.get('Ahmet'))

# 3)Add a New Element
grades['Artun'] = 90

#4) Remove an Element
grades.pop('Ahmet')

#5)Modify the value of an existing element
grades['Barkin'] = 100

#6) Search for the existence of a key
print('Ayse' in grades) #Should print "True"
print('Ahmet' in grades) #Should print "False"

#7) Search for the existence of a value
print(100 in grades.values()) #Should print "True"
print(0 in grades.values()) #Should print "False"

#8) Loop through an associative array (with foo() function)
def foo(lst):
    for i in lst:
        print(i, lst[i])
#Calling foo
foo(grades)
```

1.2) Output:

```
95
65
True
False
True
False
Barkin 100
Ayse 78
Dilruba 100
Artun 90
```

1.3) Explanation:

-We start by initializing an associative array named “grades”. Associative arrays are usually called “dictionary” in Python. All keys are “String” objects and values are integers in “grades”. Then we access a a value from the dictionary in two different ways. First is by using square brackets “[key]” and putting the key inside it in order to get the value. The second way is by using “get(key)” function which is included for dictionaries in Python by default. We add a net item to the associative array simply by “grades[“Artun”] = 90”. Since such a key does not exists in “grades” it adds the new item with the given key “Artun” and assigned value “90”. After that we remove an item from the array by using pop() function which removes the item with the given key (“Ahmet”) and returns it. Then we modify a value with the same syntax as adding an item: “grades[“Barkin”] = 100”. However, this time an item with the key “Barkin” already is in the array. Thus, instead of creating (adding) a new item we modify the item by changing its value. Next, we search for items both by keys and values. In order to do that we use “in” reserved key word which simply checks whether the given argument is in the array or not. “Ayse” is in our array so it returns “true”. Since we removed “Ahmet” it returns false. Similarly the value “100” is in the array so it returns “true” and checking whether the value “0” is in grades returns “false”. Then we declare a function called “foo()” which iterate through all the elements of our array using a “for” loop. It iterates for every item “i” inside the array and prints it key and value. We put our “grades” dictionary as the argument in order print items in “grades” one by one.

2) Dart

2.1) Code:

```
void main() {
  //1) Initialize
  var grades = {'Barkin': 95, 'Ahmet': 65, 'Ayse': 78, 'Dilruba': 100};

  //2) Get the value for a given key
  print(grades['Barkin']);

  //3) Add a new element
  grades['Artun'] = 90;

  //4) Remove an element
  grades.remove('Ahmet');

  //5) Modify the value of an existing element
  grades['Barkin'] = 100;

  //6) Search for the existence of a key
  print(grades.containsKey('Barkin'));//Should print "true"
  print(grades.containsKey('Ahmet'));//Should print "false"

  // 7)Search for the existence of a value
  print(grades.containsValue(30));//Should print "false"
  print(grades.containsValue(78));//Should print "true"

  //8) Loop through an associative array
  //Declaring function
  void foo(lst){
    lst.forEach((k, v) => print("$k: $v"));
  }
```

```
  //Calling foo
  foo(grades);
}
```

2.2) Output:

```
95
true
false
false
true
Barkin: 100
Ayse: 78
Dilruba: 100
Artun: 90
```

2.3) Explanation:

-Again we create our “grades” array. We do accessing, adding and removing items just like Python. However, this time using “remove()” function in order to remove instead of “pop()” which we used in Python. “remove()” just removes the item with the given key and does not return the item. We use “containsKey()” and “containsValue()” functions to see if an item with the given key or given value exists in the array. Then we create a function called “foo()” which iterates through all the items of its argument. The iteration is done by “forEach()” function which is also a default (built-in) function of Dart. By passing the “grades” in “foo()” function we print each key value pair one by one.

2) JavaScript

3.1) Code:

JavaScript:

```
//1) Initialize
var grades = {'Barkin': 95, 'Ahmet': 65, 'Ayse': 78, 'Dilruba': 100};

//2)Get the value for a given key
console.log(grades['Barkin']);

//3) Add a new element
grades['Artun'] = 90;

//4) Remove an element
delete grades['Ahmet'];

//5) Modify the value of an existing element
grades['Barkin'] = 100;

//6) Search for the existence of a key
console.log('Barkin' in grades);//Should print "true"
console.log('Ahmet' in grades);//Should print "false"

//7) Search for the existence of a value
var x = 30;
var y = 100;
```

```
var xb = false, yb = false;

//Iterate through the array
for(var i in grades){
   if(grades[i] == y)
      xy = true;
   if(grades[i] == x)
      xb = true;
}
console.log("Is " + x + " in grades: " + xb);
console.log("Is " + y + " in grades: " + xy);

//8) Loop through an associative array,i
function foo(lst){
   for(var i in lst){
      console.log(i + ": " + lst[i])
   }
}
foo(grades);
```

3.2) Output:

```
95
true
false
Is 30 in grades: false
Is 100 in grades: true
Barkin: 100
Ayse: 78
Dilruba: 100
Artun: 90
```

3.3) Explanation:

-Once again we create our associative array named “grades” by initializing it with the key value pairs from the previous examples. Accessing an element and removing an element is done exactly like what we did in Python and Dart. For removing an item we use the reserved word “delete” which removes the item with the key “Ahmet” from the array (“delete grades[“Ahmet]”). We check whether an element with the given key exists in the array by using “in” reserved word just like in Python. However, checking whether a value exists in the array does not work with “in” keyword in JavaScript. For this reason we iterate through the array by using “in” for each key access the values for the given keys and check whether a value is in the array or not one by one. For example “30” is a value which is not inside the array so we print that it is not inside. Then we create a “foo()” function which iterates through the array just like what we did in order to check if a value exists. Then we put our “grades” as an argument to that “foo()” function and print key-value pairs one by one.

4) Lua:

4.1) Code:

```
--1) Initialize
grades = {['Barkin'] = 95, ['Ahmet'] = 65, ['Ayse'] = 78, ['Dilruba'] = 100}
```

```
--2) Get the value for a given key
print(grades["Barkin"])
print(grades.Ahmet)

--3) Add a new element
grades["Artun"] = 90

--4) Remove an element
grades["Ahmet"] = nil

--5) Modify the value of an existing element
grades["Barkin"] = 100

--6) Search for the existence of a key
--Search by iterating over keys
x = 'Barkin'
y = 'Ahmet'
xb = false
yb = false
for k in pairs(grades) do
    if k == x then
        xb = true
    end
    if k == y then
        yb = true
    end
end

print("Is", x, "in grades:", xb)--true
print("Is", y, "in grades:", yb)--false

--7) Search for the existence of a value
val1 = 78
val2 = 30
b_val1 = false
b_val2 = false
for k, v in pairs(grades) do
    if v == val1 then
        b_val1 = true
    end
    if v == val_2 then
        b_val2 = true
    end
end

print("Is", val1, "in grades:", b_val1)--true
print("Is", val2, "in grades:", b_val2)--false

--8) Loop through an associative array
function foo(lst)
    for k, v in pairs(lst) do
    print(k, v)
```

```
    end
end
foo(grades)
```

4.1) Output:

4.1) Explanation:

-Once again we declate our associative array named “grades” and initialize it with the key-value pairs same as the previous examples. Syntax for initializing the array is similar to Python, Dart and JavaScript but we put the keys inside square brackets while initializing. We can access an item once again by using square brackets and typing the key inside the brackets (grades[“Barkin”]). However, Lua also allows accessing an item from the array, like accessing a property of an object, by using dot “.” operation (“grades.Barkin”). We add a new item to the array or modify an item just like Python (we add “Artun” and modify the value of “Barkin” in the example). For removing an element however, we access the item we want to remove and similar to modifying it we set I to “nil” (grades[“Ahmet] = nil). This operation causes the item with given key to be removed from the array instead of keeping the item with its value “nil” (null). We use “in” reserved word with the function “pairs()” in order to search whether a given key or value exists in the array. This is basically an iteration through all elements by their keys. The function “foo()” also iterates through the elements in the same way and prints key and values as pairs for each item one by one.

5) PHP:

5.1) Code:

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "<br>";
#1) Initialize
$grades = array("Barkin" => 95, "Ahmet" => 65, "Ayse" => 78, "Dilruba" => 100);

#2) Get the value for a given key
echo "Barkin's grade is $grades[Barkin]";
echo "<br>";

#3) Add a new element
$grades['Artun'] = 90;

#4) Remove an element
unset($grades['Ahmet']);

#5) Modify the value of an existing element
$grades['Barkin'] = 100;

#6) Search for the existence of a key
if(array_key_exists('Barkin', $grades)){
        echo "Barkin is in grades";
   echo "<br>";
}
```

```
else{
        echo "Barkin is not in grades";
   echo "<br>";
}

if(array_key_exists('Ahmet', $grades)){
        echo "Ahmet is in grades";
   echo "<br>";
}

else{
        echo "Ahmet is not in grades";
   echo "<br>";
}

#7) Search for the existence of a value
$x = 78;
$y = 30;
$is_x = "No";
$is_y = "No";
foreach($grades as $key => $val){
        if($val == $x)
        $is_x = "Yes";
   if($val == $y)
        $is_y = "Yes";
}
echo "Is $x in grades: $is_x";
echo "<br>";
echo "Is $y in grades: $is_y";
echo "<br>";

#8) Loop through an associative array
function foo($lst){
        foreach($lst as $key => $val){
                echo "Key: $key, Value: $val";
        echo "<br>";
        }
}
foo($grades);
?>

</body>
</html>
```

5.2) Output: (As a Text in Webpage)

Barkin's grade is 95
Barkin is in grades
Ahmet is not in grades
Is 78 in grades: Yes
Is 30 in grades: No
Key: Barkin, Value: 100

Key: Ayse, Value: 78
Key: Dilruba, Value: 100
Key: Artun, Value: 90

5. 3) Explanation:

-For this program online PHP compiler of “w3schools” is used. Since PHP is mainly used for web development the there are HTML lines at the start and end of the code segment. The HTML part was presented by the online compiler and I wrote the code segment between “<?php and ?>” which is the actual PHP program. Once again we declare an associative array named “grades” and initialize it similar to Python (also Dart, JavaScript and Lua). The syntax for initializing is just a bit different from the other languages. We use “=>” operator to match the keys with it values. Accessing, adding , removing and modifying is once again works very similar to the previous examples. We use “$grades[“Barkin”]” to access an element. If we use the same syntax for accessing an elements and assign a value to it we end up modifying it if an element with such key already exists. If the element with that key does not exists then we will end up adding a new element to the array. Removing is done by the built-in function for associative arrays called “unset()” which removes the item with the given key (removes “Ahmet” in the example). We check whether an item exists with a key by using the function “array_key_exists()” which takes the key as an argument and returns a boolean (“true” if exists, “false” if not). Since there is no built-in function that checks whether a value is in the array, I used “foreach()” method which iterates over each element and than for each element I checked if a value exists or not by accessing its value (similar to what we did in Lua and JacaScript). Then, we created a function named “foo()” which uses the same iteration technique over its argument. By passing “grades” in to the function we printed each key-value pair one by one.

6) Ruby:

6.1) Code:

```
#1) Initialize
grades = {"Barkin" => 95, "Ahmet" => 65, "Ayse" => 78, "Dilruba" => 100}

#2) Get the value for a given key
puts 'Barkins grade is ' + grades['Barkin'].to_s

#3)Add a new element
grades['Artun'] = 90

#4)Remove an element
grades.delete('Ahmet')

#5) Modify the value of an existing element
grades['Barkin'] = 100

#6) Search for the existence of a key
puts grades.key?('Barkin') #Should print "true"
puts grades.key?('Ahmet') #Should print "false"

#7) Search for the existence of a value
puts grades.value?(78) #Should print "true"
puts grades.value?(30) #Should print "false"
```

```
#8)Loop through an associative array
def foo(lst)
  lst.each do |k, v|
    puts "Name: #{k}, Grade: #{v}"
  end
end
foo(grades)
```

6.2) Output:

```
Barkins grade is 95
true
false
true
false
Name: Barkin, Grade: 100
Name: Ayse, Grade: 78
Name: Dilruba, Grade: 100
Name: Artun, Grade: 90
```

6.3) Explanation:

-Once again we declare and initialize an associative array called “grades”. Initialization is done similar to the other example (especially PHP since we use the “=>” operator). Once again we access, add, remove or modify an element just like Python with minor syntactical differences (once again we add “Artun”, remove “Ahmet”, change value of “Barkin”). We use the built-in function to check whether an item with a given key (“grades.key?()”) or given value (“grades.value?()”) exists. Lastly, we define a function called “foo()” which iterates over its argument by using built-in function “each” with the following syntax “lst.each do |k, v|”. “k” and “v” is assigned to key and value of the current item during iteration. We print the “k”, and “v” variables which are basically key-value pairs for each item one by one by passing our “grades” array in to the “foo()” function.

7) Rust:

7.1) Code:

```
use std::collections::HashMap;
fn main() {
   //1) Initialize
   let mut grades: HashMap<&str, i32> = [("Barkin", 95), ("Ahmet", 65), ("Ayse", 78), ("Dilruba",
100)].into_iter().collect();

   //2) Get the value for a given key
   println!("Barkin has the grade: {:?}", grades["Barkin"]);

   //3) Add a new element
   grades.insert( "Artun", 90 );

   //4) Remove an element
   grades.remove("Ahmet");

   //5)Modify the value of an existing element
    grades.insert( "Barkin", 100 );
```

```
    //6) Search for the existence of a key
    println!("{}", grades.contains_key("Barkin")); //Should print "true"
    println!("{}", grades.contains_key("Ahmet")); //Should print "false"

    //7) Search for the existence of a value
    let x = 100 as i32;
    let y = 30 as i32;

    let bool_x = grades.values().any(|v| v ==  &x );
    let bool_y = grades.values().any(|v| v ==  &y );
    println!("{:?}", bool_x); //Should print "true"
    println!("{:?}", bool_y); //Should print "false"

    //8) Loop through an associative array
    for(k, v) in &grades{
      println!("{}: {}", k, v);
    }
}
```

7.2) Output:

7.3) Explanation:

-First we import the “HashMap” class from “collections” which is in the default library Once again we create an instance of an associative array (“HashMap”) called “grades” and initialize it. Initialization of the map is rather different than other languages in this homework. We declare the type of the keys and values which is not necessary for other languages. We access an element similar to Python (grades[“Barkin”]) which is by the key inside square brackets. Modifying an element and adding an element has the same logic with the other languages. However, we use a function called “insert()” which comes with the “HashMap” class. If a key of the given name (“Artun” in our example) is not in the map than it is added to the map. If it already exists it is modified. However, I could not find information about the background logic of modification. It is possible that when we try to “insert” an item with an existing key what the program actually does is to remove the item and add it back with the new value. I also could not test if that is the case because while printing the “grades” one by one iteration order seems to change for different executions. I am not sure whether this is caused by the compiler I used or caused by how iteration works in Rust. For these reasons Rust was the only language in this homework which seemed rather unclear to me in terms of associative arrays. Also another fact to consider about associative arrays in Rust is that while all the other languages are directly supporting associative arrays for Rust we imported the “HashMap” class since Rust does not directly allows creation of associative arrays like Python’s dictionaries. Also, the type for keys and values must be declared which makes me think of the maps as “hash maps” and not as “associative arrays” unlike the languages. Removing is also done by a function which is called “remove()” and it removes the item with the given key from the map. For checking whether a given key exists in the map we used the function named as “contains_key()”. Checking whether a given value exists in the map is also done by a function which is called “values()”. Lastly, we iterated over each element by using the reserved word “in” similar to Python. In order to do that we declared a tuple “(k,v)” which basically get the key and value for each elements Then, we printed “k an v” (key-value pairs) one by one.