Muhammed İkbal Doğan

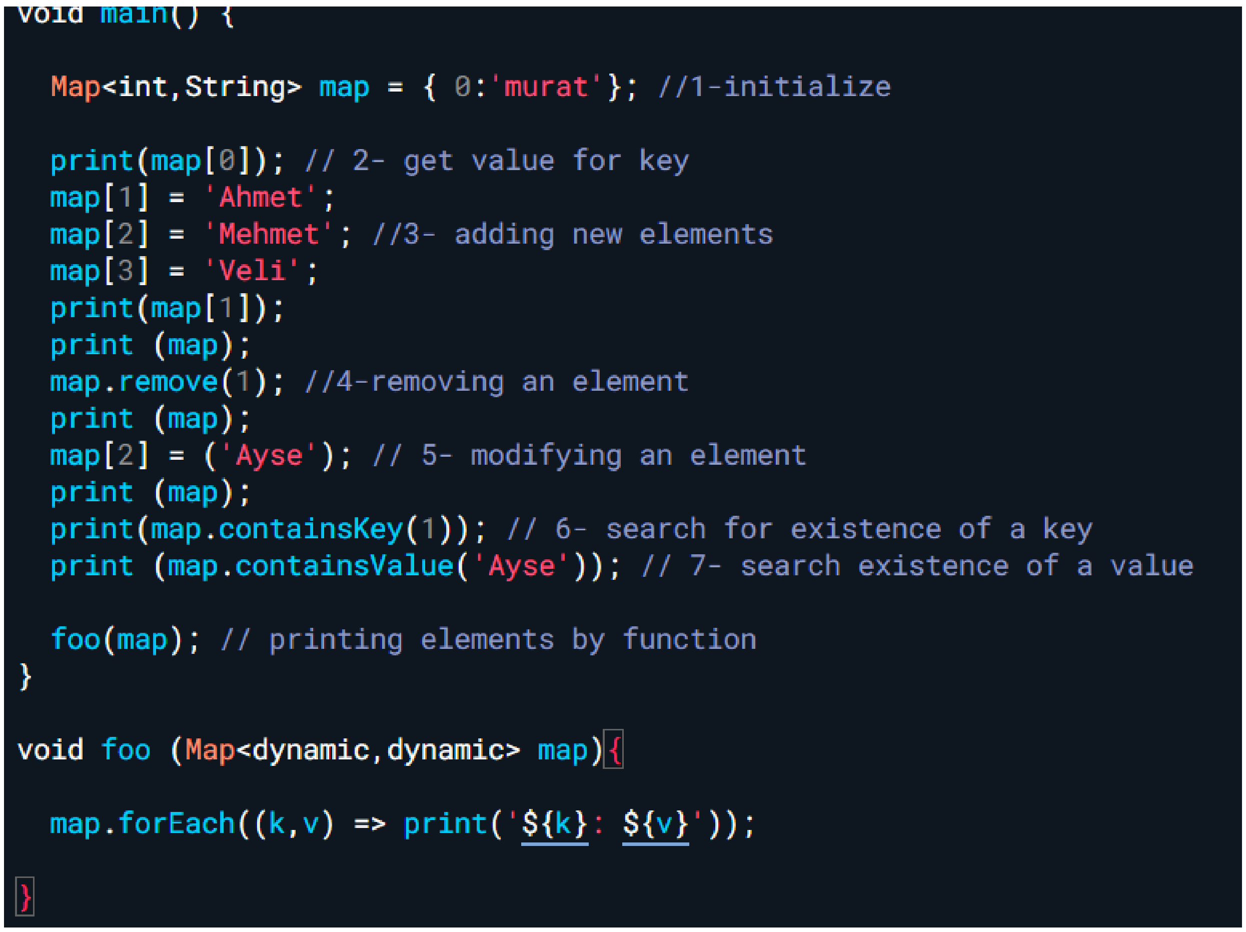
21702990

CS315-03

Homework 1

DART

Code



```
void main() {

  Map<int,String> map = { 0:'murat'}; //1-initialize

  print(map[0]); // 2- get value for key
  map[1] = 'Ahmet';
  map[2] = 'Mehmet'; //3- adding new elements
  map[3] = 'Veli';
  print(map[1]);
  print (map);
  map.remove(1); //4-removing an element
  print (map);
  map[2] = ('Ayse'); // 5- modifying an element
  print (map);
  print(map.containsKey(1)); // 6- search for existence of a key
  print (map.containsValue('Ayse')); // 7- search existence of a value

  foo(map); // printing elements by function
}

void foo (Map<dynamic,dynamic> map){

  map.forEach((k,v) => print('${k}: ${v}'));

}
```

void main() {

  Map<int,String> map = { 0:'murat'}; //1-initialize

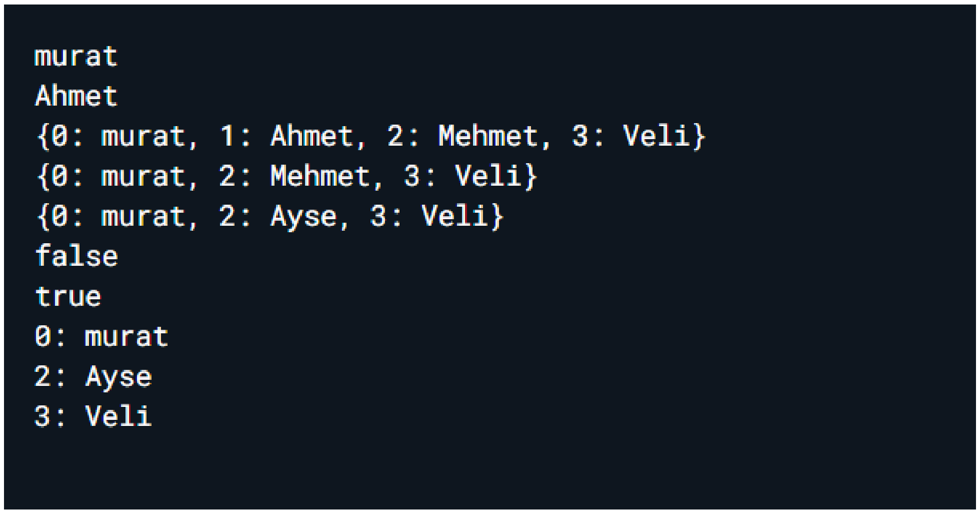
  print(map[0]); // 2- get value for key

  map[1] = 'Ahmet';

  map[2] = 'Mehmet'; //3- adding new elements

```
  map[3] = 'Veli';
  print(map[1]);
  print (map);
  map.remove(1); //4-removing an element
  print (map);
  map[2] = ('Ayse'); // 5- modifying an element
  print (map);
  print(map.containsKey(1)); // 6- search for existence of a key
  print (map.containsValue('Ayse')); // 7- search existence of a value

  foo(map); // printing elements by function
}

void foo (Map<dynamic,dynamic> map){

  map.forEach((k,v) => print('${k}: ${v}'));

}
```

OUTPUT

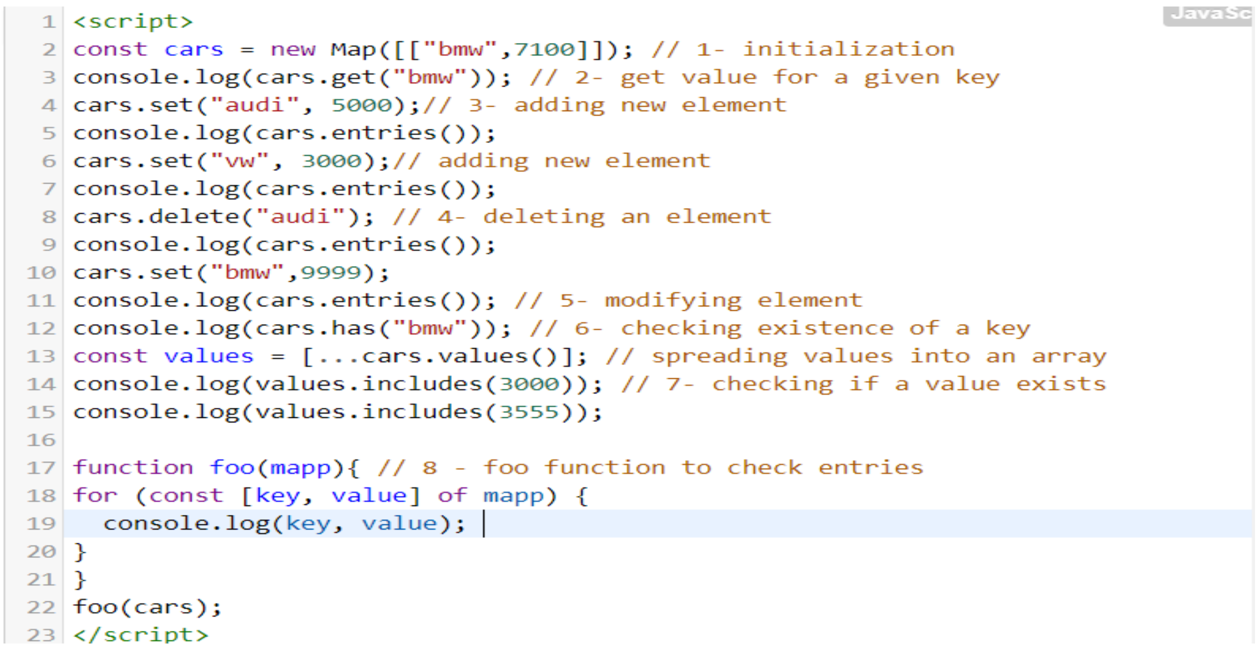


```
murat
Ahmet
{0: murat, 1: Ahmet, 2: Mehmet, 3: Veli}
{0: murat, 2: Mehmet, 3: Veli}
{0: murat, 2: Ayse, 3: Veli}
false
true
0: murat
2: Ayse
3: Veli
```

Firstly I initialized the map with initial key-value which is 0 and murat. Then get the value for key 0 which outputs ‘murat’. Added some key value pairs and print the whole map. After that I removed the key 1 which has value Ahmet and it’s removed on output.Then I modified the key 2’s value from ‘Mehmet’ to ‘Ayse. Then checked if the map has key 1 but it returned false because it has been deleted. Searched existence of value “Ayse” and it returned true. Lastly I wrote a foo function which printed the whole key value pair map.

JAVASCRIPT

Code

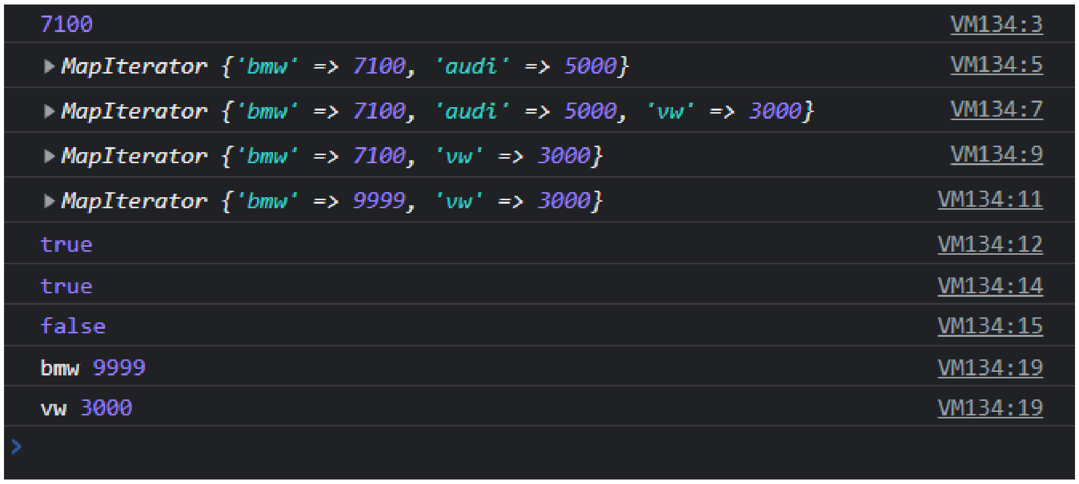


```
<script>
const cars = new Map([["bmw",7100]]); // 1- initialization
console.log(cars.get("bmw")); // 2- get value for a given key
cars.set("audi", 5000);// 3- adding new element
console.log(cars.entries());
cars.set("vw", 3000);// adding new element
console.log(cars.entries());
cars.delete("audi"); // 4- deleting an element
console.log(cars.entries());
cars.set("bmw",9999);
console.log(cars.entries()); // 5- modifying element
console.log(cars.has("bmw")); // 6- checking existence of a key
const values = [...cars.values()]; // spreading values into an array
console.log(values.includes(3000)); // 7- checking if a value exists
console.log(values.includes(3555));

function foo(mapp){ // 8 - foo function to check entries
for (const [key, value] of mapp) {
  console.log(key, value); 
}
}
foo(cars);
</script>
```

```
<script>

const cars = new Map([["bmw",7100]]); // 1- initialization

console.log(cars.get("bmw")); // 2- get value for a given key

cars.set("audi", 5000);// 3- adding new element

console.log(cars.entries());

cars.set("vw", 3000);// adding new element

console.log(cars.entries());

cars.delete("audi"); // 4- deleting an element

console.log(cars.entries());

cars.set("bmw",9999);
```

```
console.log(cars.entries()); // 5- modifying element

console.log(cars.has("bmw")); // 6- checking existence of a key

const values = [...cars.values()]; // spreading values into an array

console.log(values.includes(3000)); // 7- checking if a value exists

console.log(values.includes(3555));


function foo(mapp){ // 8 - foo function to check entries

for (const [key, value] of mapp) {

  console.log(key, value);

}

}

foo(cars);

</script>
```

OUTPUT

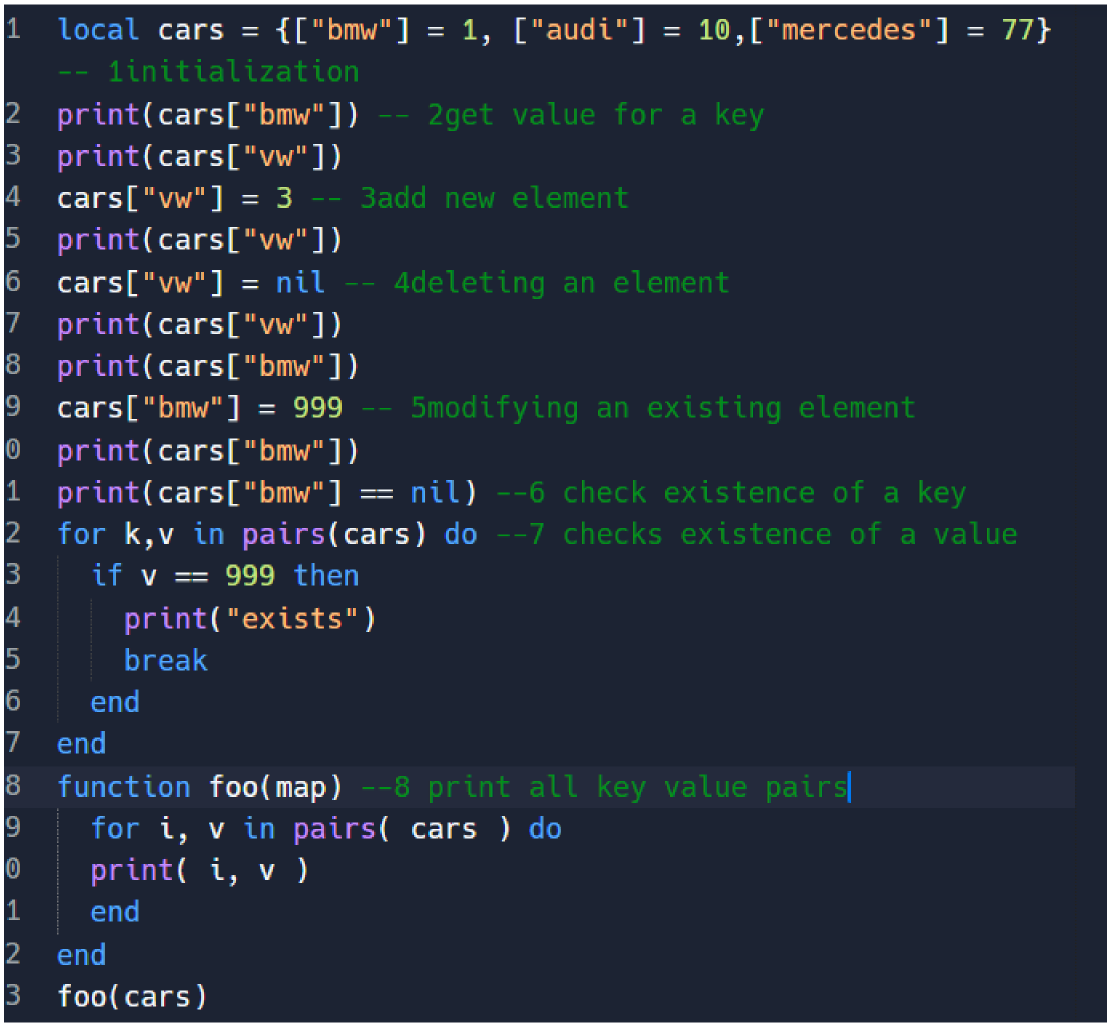


```
7100                                                              VM134:3
▶MapIterator {'bmw' => 7100, 'audi' => 5000}                      VM134:5
▶MapIterator {'bmw' => 7100, 'audi' => 5000, 'vw' => 3000}        VM134:7
▶MapIterator {'bmw' => 7100, 'vw' => 3000}                        VM134:9
▶MapIterator {'bmw' => 9999, 'vw' => 3000}                        VM134:11
true                                                              VM134:12
true                                                              VM134:14
false                                                             VM134:15
bmw 9999                                                          VM134:19
vw 3000                                                           VM134:19
>
```

Map is initialized with initial key-value pair which is “bmw” and 7100. Then value of key “bmw” printed. New element is added and map is printed. Then “audi” – 5000 key value pair is deleted and map is printed. Then key “bmw”’s value is modified from 7100 to 9999.Then I spread values into an array and checked if values exists and outputs are shown . Lastly, I printed the whole map.

LUA

Code



```lua
local cars = {["bmw"] = 1, ["audi"] = 10,["mercedes"] = 77}
-- 1initialization
print(cars["bmw"]) -- 2get value for a key
print(cars["vw"])
cars["vw"] = 3 -- 3add new element
print(cars["vw"])
cars["vw"] = nil -- 4deleting an element
print(cars["vw"])
print(cars["bmw"])
cars["bmw"] = 999 -- 5modifying an existing element
print(cars["bmw"])
print(cars["bmw"] == nil) --6 check existence of a key
for k,v in pairs(cars) do --7 checks existence of a value
  if v == 999 then
    print("exists")
    break
  end
end
function foo(map) --8 print all key value pairs
  for i, v in pairs( cars ) do
  print( i, v )
  end
end
foo(cars)
```

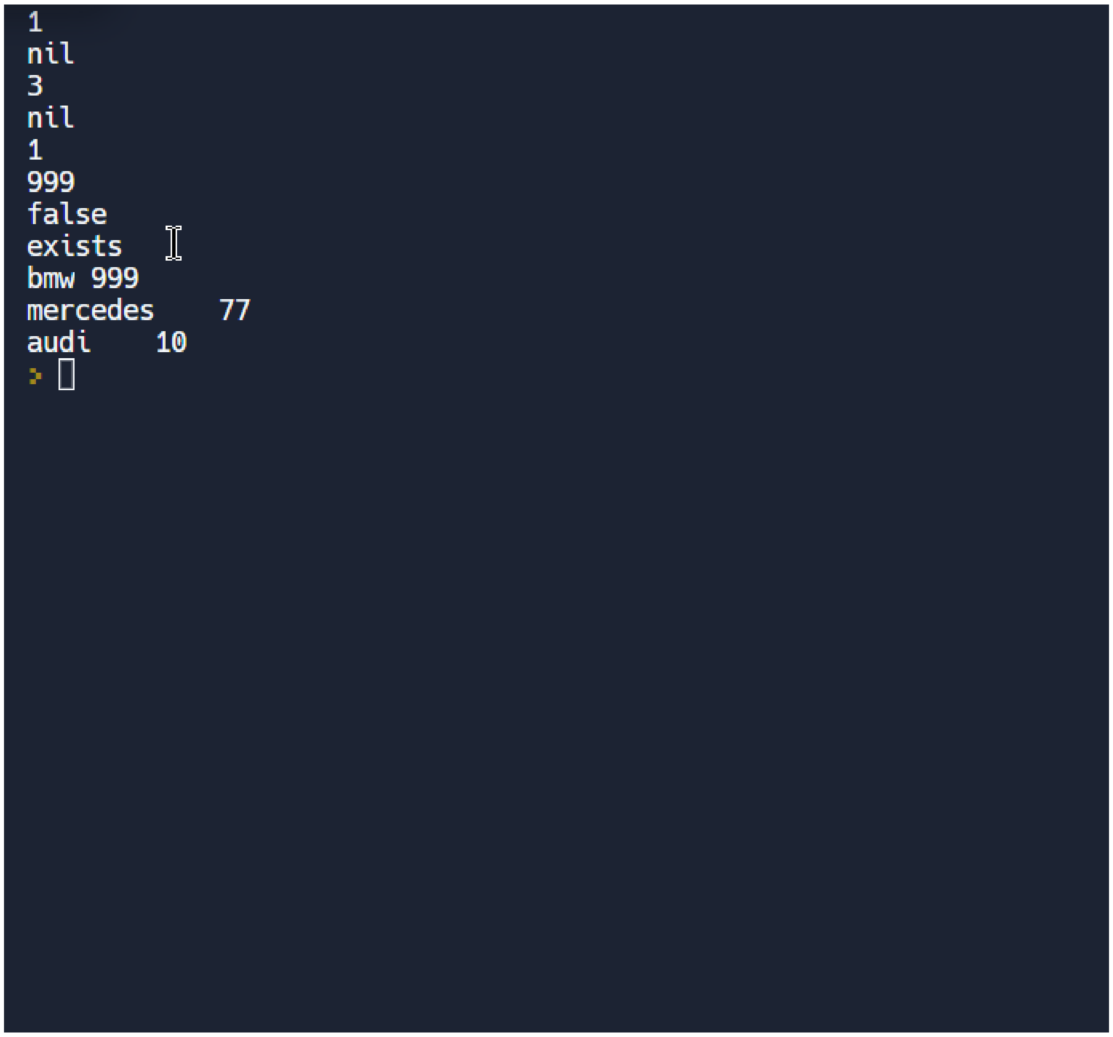
local cars = {["bmw"] = 1, ["audi"] = 10,["mercedes"] = 77} --
1initialization

print(cars["bmw"]) -- 2get value for a key

```
print(cars["vw"])

cars["vw"] = 3 -- 3add new element

print(cars["vw"])

cars["vw"] = nil -- 4deleting an element

print(cars["vw"])

print(cars["bmw"])

cars["bmw"] = 999 -- 5modifying an existing element

print(cars["bmw"])

print(cars["bmw"] == nil) --6 check existence of a key

for k,v in pairs(cars) do --7 checks existence of a value

  if v == 999 then

    print("exists")

    break

  end

end

function foo(map) --8 print all key value pairs

  for i, v in pairs( cars ) do

  print( i, v )

  end

end
```

foo(cars)

OUTPUT

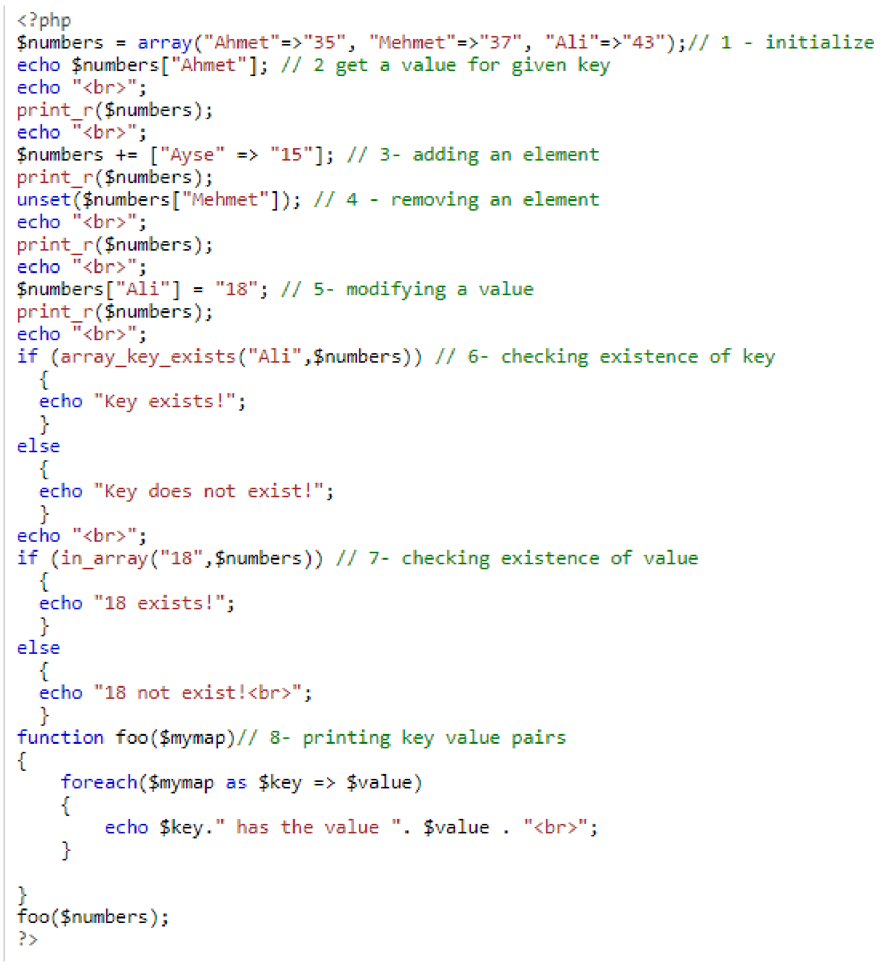


```
1
nil
3
nil
1
999
false
exists
bmw 999
mercedes    77
audi    10
>
```

Firstly I initialized map with 3 different key-value pairs. Then get the value of key “bmw” which is 1. Then added “vw”-3 key-value pair. Deleted “vw” by make it equal to nil. As it seen before I added “vw” it was nil so deleting it means making it equal to nil. Then I modified “bmw”’s value from 1 to 999 and printed it. Checked existence of “bmw” in if condition that if it is equal to nil but it returned false so it means it exists. Then in a for loop I checked all values of keys and if value 999 exists it prints “exists”. Lastly I printed whole map in foo function.

PHP

Code


```
<?php
$numbers = array("Ahmet"=>"35", "Mehmet"=>"37", "Ali"=>"43");// 1 - initialize
echo $numbers["Ahmet"]; // 2 get a value for given key
echo "<br>";
print_r($numbers);
echo "<br>";
$numbers += ["Ayse" => "15"]; // 3- adding an element
print_r($numbers);
unset($numbers["Mehmet"]); // 4 - removing an element
echo "<br>";
print_r($numbers);
echo "<br>";
$numbers["Ali"] = "18"; // 5- modifying a value
print_r($numbers);
echo "<br>";
if (array_key_exists("Ali",$numbers)) // 6- checking existence of key
  {
  echo "Key exists!";
  }
else
  {
  echo "Key does not exist!";
  }
echo "<br>";
if (in_array("18",$numbers)) // 7- checking existence of value
  {
  echo "18 exists!";
  }
else
  {
  echo "18 not exist!<br>";
  }
function foo($mymap)// 8- printing key value pairs
{
    foreach($mymap as $key => $value)
    {
        echo $key." has the value ". $value . "<br>";
    }

}
foo($numbers);
?>
```

<?php

$numbers = array("Ahmet"=>"35", "Mehmet"=>"37", "Ali"=>"43");// 1 - initialize

echo $numbers["Ahmet"]; // 2 get a value for given key

echo "<br>";

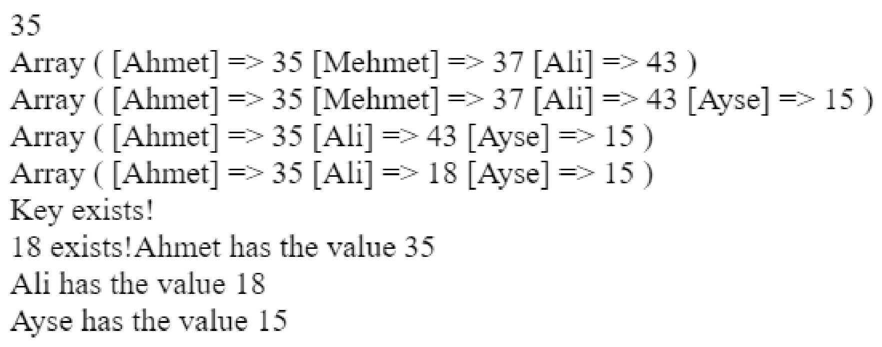
print_r($numbers);

echo "<br>";

```
$numbers += ["Ayse" => "15"]; // 3- adding an element

print_r($numbers);

unset($numbers["Mehmet"]); // 4 - removing an element

echo "<br>";

print_r($numbers);

echo "<br>";

$numbers["Ali"] = "18"; // 5- modifying a value

print_r($numbers);

echo "<br>";

if (array_key_exists("Ali",$numbers)) // 6- checking existence of key

  {

  echo "Key exists!";

  }

else

  {

  echo "Key does not exist!";

  }

echo "<br>";

if (in_array("18",$numbers)) // 7- checking existence of value

  {
```

```
  echo "18 exists!";

  }

else

  {

  echo "18 not exist!<br>";

  }

function foo($mymap)// 8- printing key value pairs

{

        foreach($mymap as $key => $value)

        {

                echo $key." has the value ". $value . "<br>";

        }


}

foo($numbers);

?>
```

OUTPUT

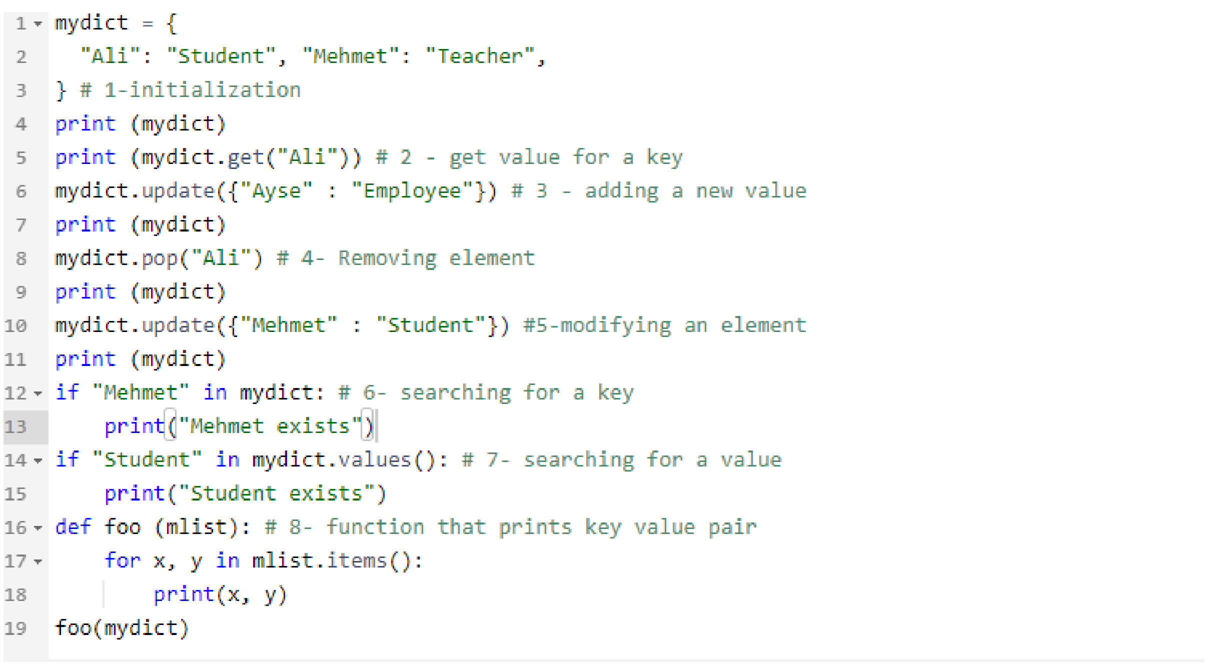


```
35
Array ( [Ahmet] => 35 [Mehmet] => 37 [Ali] => 43 )
Array ( [Ahmet] => 35 [Mehmet] => 37 [Ali] => 43 [Ayse] => 15 )
Array ( [Ahmet] => 35 [Ali] => 43 [Ayse] => 15 )
Array ( [Ahmet] => 35 [Ali] => 18 [Ayse] => 15 )
Key exists!
18 exists!Ahmet has the value 35
Ali has the value 18
Ayse has the value 15
```

Firstly I initialize a map with 3 initial key value pairs. Then get the value of “Ahmet” which is 35.Then added “Ayse” – 15 key value pair and printed. Removed “Mehmet” -37 key-value pair and printed the map. Modified “Ali”’s value from 43 to 18. Then checked existence of key”Ali” in if and printed. Then checked existence of key 18 and printed. Lastly I printed the whole map.

PYTHON

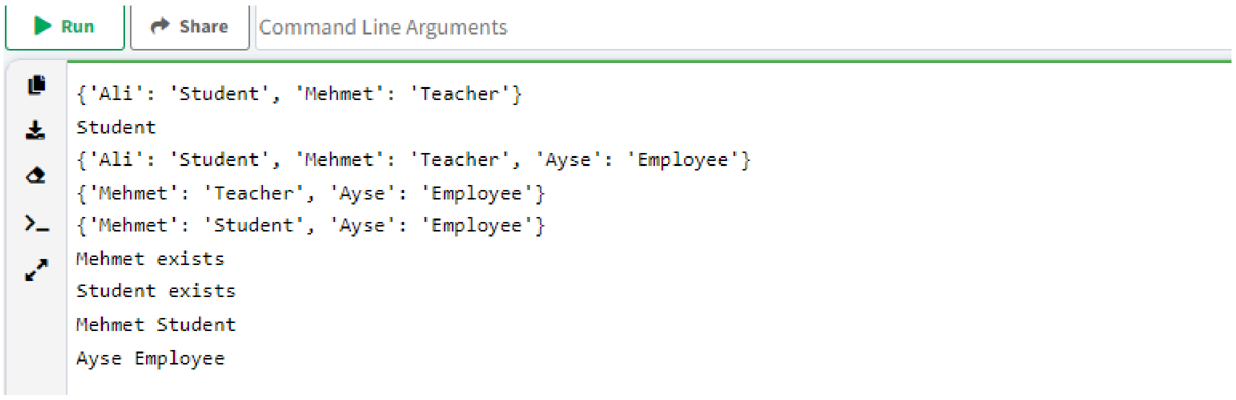
Code



```python
mydict = {
  "Ali": "Student", "Mehmet": "Teacher",
} # 1-initialization
print (mydict)
print (mydict.get("Ali")) # 2 - get value for a key
mydict.update({"Ayse" : "Employee"}) # 3 - adding a new value
print (mydict)
mydict.pop("Ali") # 4- Removing element
print (mydict)
mydict.update({"Mehmet" : "Student"}) #5-modifying an element
print (mydict)
if "Mehmet" in mydict: # 6- searching for a key
    print("Mehmet exists")
if "Student" in mydict.values(): # 7- searching for a value
    print("Student exists")
def foo (mlist): # 8- function that prints key value pair
    for x, y in mlist.items():
        print(x, y)
foo(mydict)
```

```
mydict = {
  "Ali": "Student", "Mehmet": "Teacher",
} # 1-initialization
print (mydict)
print (mydict.get("Ali")) # 2 - get value for a key
mydict.update({"Ayse" : "Employee"}) # 3 - adding a new value
print (mydict)
mydict.pop("Ali") # 4- Removing element
print (mydict)
mydict.update({"Mehmet" : "Student"}) #5-modifying an element
print (mydict)
if "Mehmet" in mydict: # 6- searching for a key
    print("Mehmet exists")
if "Student" in mydict.values(): # 7- searching for a value
    print("Student exists")
def foo (mlist): # 8- function that prints key value pair
    for x, y in mlist.items():
        print(x, y)
foo(mydict)
```

OUTPUT

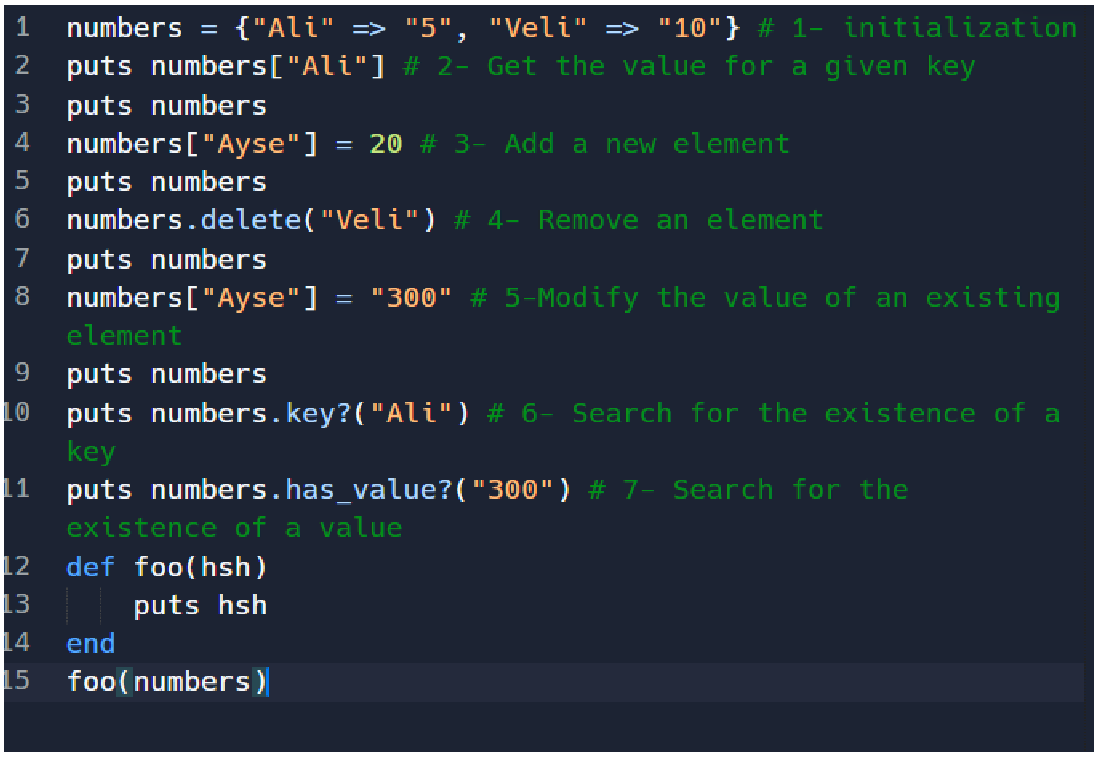


```
{'Ali': 'Student', 'Mehmet': 'Teacher'}
Student
{'Ali': 'Student', 'Mehmet': 'Teacher', 'Ayse': 'Employee'}
{'Mehmet': 'Teacher', 'Ayse': 'Employee'}
{'Mehmet': 'Student', 'Ayse': 'Employee'}
Mehmet exists
Student exists
Mehmet Student
Ayse Employee
```

Firstly I initialized a dictionary with 2 key-value pairs. Then printed value for key "Ali" which is "Student". Then added "Ayse"-"Employee" key –value pair and printed the map. Then removed "Ali" –"Student" key value pair and printed. Then modified "Mehmet" key's value from "Teacher" to "Student". Then checked for existence of key "Mehmet" in if also checked "Student" value in if. Lastly I printed to whole map.

RUBY

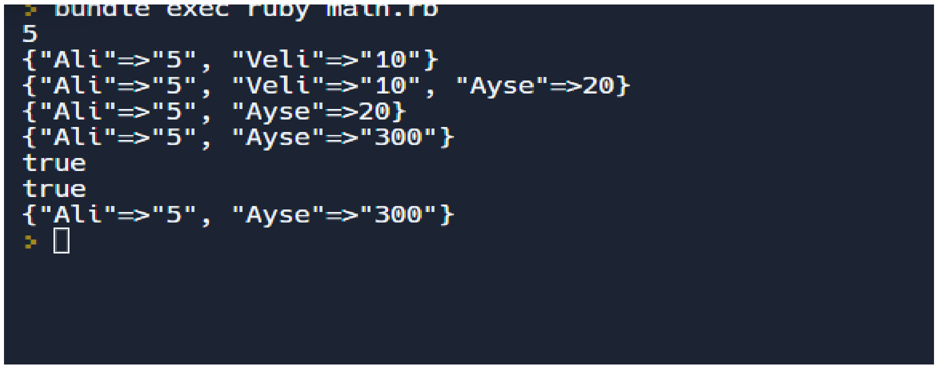
Code



```ruby
numbers = {"Ali" => "5", "Veli" => "10"} # 1- initialization
puts numbers["Ali"] # 2- Get the value for a given key
puts numbers
numbers["Ayse"] = 20 # 3- Add a new element
puts numbers
numbers.delete("Veli") # 4- Remove an element
puts numbers
numbers["Ayse"] = "300" # 5-Modify the value of an existing element
puts numbers
puts numbers.key?("Ali") # 6- Search for the existence of a key
puts numbers.has_value?("300") # 7- Search for the existence of a value
def foo(hsh)
    puts hsh
end
foo(numbers)
```

```
numbers = {"Ali" => "5", "Veli" => "10"} # 1- initialization

puts numbers["Ali"] # 2- Get the value for a given key

puts numbers

numbers["Ayse"] = 20 # 3- Add a new element

puts numbers

numbers.delete("Veli") # 4- Remove an element

puts numbers

numbers["Ayse"] = "300" # 5-Modify the value of an existing element

puts numbers

puts numbers.key?("Ali") # 6- Search for the existence of a key

puts numbers.has_value?("300") # 7- Search for the existence of a value

def foo(hsh)

  puts hsh

end

foo(numbers)
```

OUTPUT

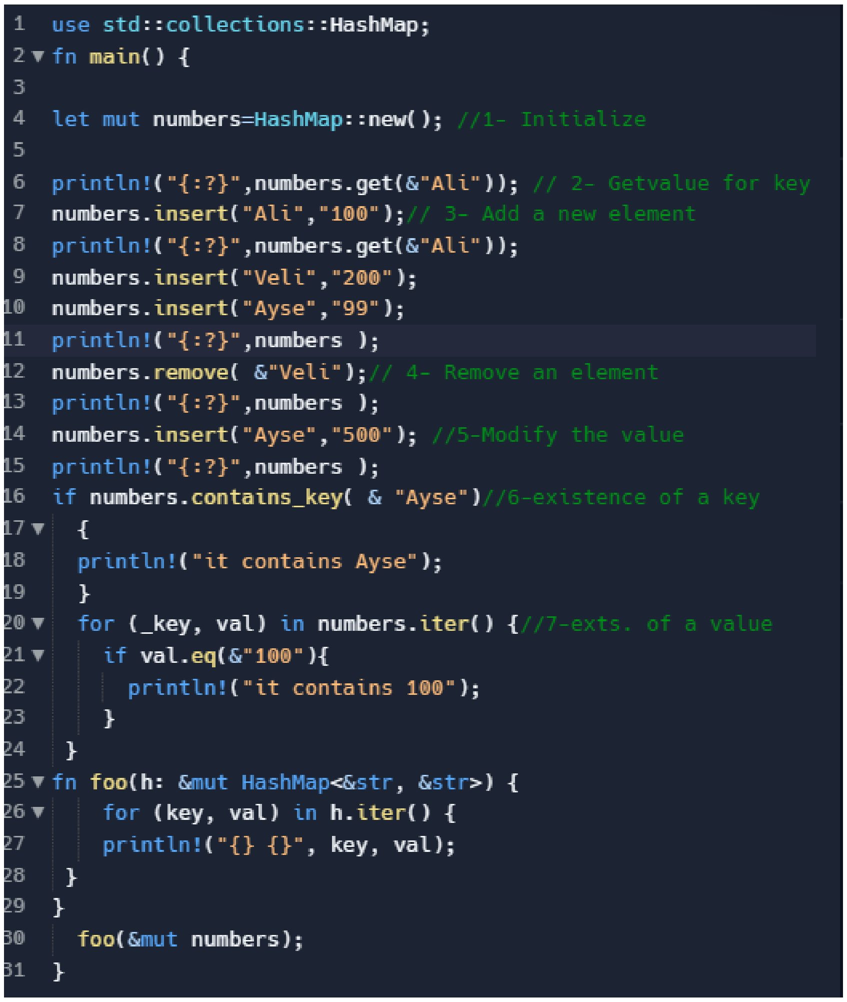


```
> bundle exec ruby main.rb
5
{"Ali"=>"5", "Veli"=>"10"}
{"Ali"=>"5", "Veli"=>"10", "Ayse"=>20}
{"Ali"=>"5", "Ayse"=>20}
{"Ali"=>"5", "Ayse"=>"300"}
true
true
{"Ali"=>"5", "Ayse"=>"300"}
> 
```

Firstly I initialized map with 2 key value pairs then printed the value of “Ali” which is 5. Then added “Ayse” – 20 key value pair. Then deleted the “Veli” – 10 key value pair and printed the map to show changes. Then modified “Ayse”’s value from 20 to 300. Then checked if map contains key “Ali” and value “300” and printed them. Lastly I printed the whole map in function.

RUST

Code



```
use std::collections::HashMap;
fn main() {

let mut numbers=HashMap::new(); //1- Initialize

println!("{:?}",numbers.get(&"Ali")); // 2- Getvalue for key
numbers.insert("Ali","100");// 3- Add a new element
println!("{:?}",numbers.get(&"Ali"));
numbers.insert("Veli","200");
numbers.insert("Ayse","99");
println!("{:?}",numbers );
numbers.remove( &"Veli");// 4- Remove an element
println!("{:?}",numbers );
numbers.insert("Ayse","500"); //5-Modify the value
println!("{:?}",numbers );
if numbers.contains_key( & "Ayse")//6-existence of a key
  {
  println!("it contains Ayse");
  }
  for (_key, val) in numbers.iter() {//7-exts. of a value
    if val.eq(&"100"){
      println!("it contains 100");
    }
 }
fn foo(h: &mut HashMap<&str, &str>) {
    for (key, val) in h.iter() {
    println!("{} {}", key, val);
 }
}
  foo(&mut numbers);
}
```

use std::collections::HashMap;

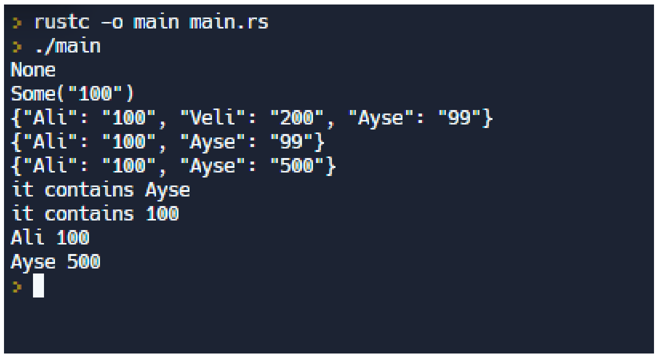
fn main() {

let mut numbers=HashMap::new(); //1- Initialize

println!("{:?}",numbers.get(&"Ali")); // 2- Getvalue for key

```
numbers.insert("Ali","100");// 3- Add a new element
println!("{:?}",numbers.get(&"Ali"));
numbers.insert("Veli","200");
numbers.insert("Ayse","99");
println!("{:?}",numbers );
numbers.remove( &"Veli");// 4- Remove an element
println!("{:?}",numbers );
numbers.insert("Ayse","500"); //5-Modify the value
println!("{:?}",numbers );
if numbers.contains_key( & "Ayse")//6-existence of a key
 {
 println!("it contains Ayse");
 }
 for (_key, val) in numbers.iter() {//7-exts. of a value
  if val.eq(&"100"){
    println!("it contains 100");
  }
}
fn foo(h: &mut HashMap<&str, &str>) {
  for (key, val) in h.iter() {
    println!("{} {}", key, val);
 }
}
  foo(&mut numbers);
}
```

OUTPUT



```
> rustc -o main main.rs
> ./main
None
Some("100")
{"Ali": "100", "Veli": "200", "Ayse": "99"}
{"Ali": "100", "Ayse": "99"}
{"Ali": "100", "Ayse": "500"}
it contains Ayse
it contains 100
Ali 100
Ayse 500
>
```

Firstly I initialize a map without initial value and searched for a key and get none value. Then I inserted the "Ali" - 100 key – value pair and printed the value. Then inserted two more key-value pairs and Removed "Veli" – 200 and printed that it is removed. Then modified Ayse's value from 99 to 500. Because the key exists, when I insert the same key with different value, it modifies the value. Then searched if "Ayse" key exists and printed. Then in for loop checked for all values and printed that value 100 exists. Lastly I printed the map in function.

DISCUSSION

Dart is easy to write and also it's easy to check if map contains key and array just with one function of map. It's also readable generally. For Javascript it's also writable but it does not have the function for checking values. It's also readable everything seem reasonable. For Lua it is comfortable to write but it also does not have function for checking value. When to remove an element we should equal it no nil which is not good practice for write maybe. For php it is not complicated to write but not as comfortable as the previous languages but it has handy functions to check key value pairs and it's readable. Python is pretty easy to write and read I think which feels most comfortable while writing. Ruby has also handy structure to write and it’s clear that what’s written while reading. Rust is the worst for readability I think and it is not comfortable that writing in rust. To sum up I think Python is best for both readability and writablity and rust is the most uncomfortable to read and write.

LEARNING STRATEGY

Firstly I have read the chapters from book then checked https://www.w3schools.com/ to learn the syntax for languages except lua. For lua I checked https://www.lua.org/ to learn it’s syntax. I used online compiler which is replit.com/languages. For map structures and I checked https://programming-idioms.org .