# CS315

## Programming Languages

## HW1

Elifsena Öz

22002245

Section 1

## Explanation of the Code

In this homework I have created associative arrays and used their functionalities. Languages involved in this homework are Dart, JavaScript, Lua, PHP, Python, Ruby and Rust. The codes basically do the same operations for each language. After each operation the array is printed. First, an associative array is created with keys and values corresponding to 4 characters from a series, “How I Met Your Mother”. Value with the key ‘robin’ is printed. A new value, ‘Lily Aldrin’ is added with key ‘lily’. Value with key ‘robin’ is deleted. Value with key ‘lily’ is changed to ‘Lily Erikson’. The array is checked for the existence of key ‘lily’. The array is checked for the existence of value ‘Ted Mosby’. Lastly the array is printed by passing the key and value as arguments for the function named foo. The code is the same for each language except adjustments in syntax.

## Dart

**1. Initialize**

```
print("1. Initialize");
var himym = {
    'marshall': 'Marshall Erikson',
    'ted': 'Ted Mosby',
    'barney': 'Barney Sitinson',
    'robin': 'Robin Sherbatsky',
};
print(himym);
```

**Output:**

```
1. Initialize
{marshall: Marshall Erikson, ted: Ted Mosby, barney: Barney Sitinson,
robin: Robin Sherbatsky}
```

**2. Get the value for given key**

```
print("\n2. Get the value for a given key");
var robin = himym['robin'];
print("element with key robin: $robin");
```

**Output:**

```
2. Get the value for a given key
element with key robin: Robin Sherbatsky
```

**3. Add a new element**

```
print("\n3. Add a new element");
himym['lily'] = 'Lily Aldrin';
print(himym);
```

**Output:**

```
3. Add a new element
{marshall: Marshall Erikson, ted: Ted Mosby, barney: Barney Sitinson,
robin: Robin Sherbatsky, lily: Lily Aldrin}
```

**4. Remove an element**

```
print("\n4. Remove an element");
himym.remove('robin');
print(himym);
```

**Output:**

```
4. Remove an element
{marshall: Marshall Erikson, ted: Ted Mosby, barney: Barney Sitinson,
lily: Lily Aldrin}
```

**5. Modify the value of an existing element**

```
print("\n5. Modify the value of an existing element");
himym['lily'] = 'Lily Erikson';
print(himym);
```

**Output:**

```
5. Modify the value of an existing element
{marshall: Marshall Erikson, ted: Ted Mosby, barney: Barney Sitinson,
lily: Lily Erikson}
```

**6. Search for the existence of a key**

```
print("\n6. Search for the existence of a key");
for (String key in himym.keys) {
    if (key == 'lily') {
    print("himym contains element with key lily");
    }
}
```

**Output:**

```
6. Search for the existence of a key
himym contains element with key lily
```

**7. Search for the existence of a value**

```
print("\n7. Search for the existence of a value");
for (String value in himym.values) {
    if (value == 'Ted Mosby') {
    print("himym contains Ted Mosby");
    }
}
```

**Output:**

```
7. Search for the existence of a value
himym contains Ted Mosby
```

**8. Print keys and values using foo function**

```
print("\n8. Print keys and values using foo");

void foo(String key){
    var value = himym[key];
    print("$key : $value");
}

for (String key in himym.keys) {
```

```
        foo(key);
    }
```

**Output:**

```
8. Print keys and values using foo
marshall : Marshall Erikson
ted : Ted Mosby
barney : Barney Sitinson
lily : Lily Erikson
```

## JavaScript

**1. Initialize**

```
print("1. Initialize");
var himym = {
    'marshall': 'Marshall Erikson',
    'ted': 'Ted Mosby',
    'barney': 'Barney Sitinson',
    'robin': 'Robin Sherbatsky',
};
print(himym);
```

**Output:**

```
Map(4) {
     'marshall' => 'Marshall Erikson',
     'ted' => 'Ted Mosby',
     'barney' => 'Barney Sitinson',
     'robin' => 'Robin Sherbatsky'
}
```

**2. Get the value for given key**

```
console.log('\n2. Get the value for a given key');
console.log('element with key robin: ' + himym.get('robin'));
```

**Output:**

```
2. Get the value for a given key
element with key robin: Robin Sherbatsky
```

**3. Add a new element**

```
console.log('\n3. Add a new element');
himym.set('lily', 'Lily Aldrin');
console.log(himym);
```

**Output:**

```
3. Add a new element
Map(5) {
     'marshall' => 'Marshall Erikson',
     'ted' => 'Ted Mosby',
     'barney' => 'Barney Sitinson',
     'robin' => 'Robin Sherbatsky',
     'lily' => 'Lily Aldrin'
```

```
}
```

**4. Remove an element**

```
console.log('\n4. Remove an element');
himym.delete('robin');
console.log(himym);
```

**Output:**

```
4. Remove an element
Map(4) {
      'marshall' => 'Marshall Erikson',
      'ted' => 'Ted Mosby',
      'barney' => 'Barney Sitinson',
      'lily' => 'Lily Aldrin'
}
```

**5. Modify the value of an existing element**

```
console.log('\n5. Modify the value of an existing element');
himym.set('lily', 'Lily Erikson');
console.log(himym);
```

**Output:**

```
5. Modify the value of an existing element
Map(4) {
      'marshall' => 'Marshall Erikson',
      'ted' => 'Ted Mosby',
      'barney' => 'Barney Sitinson',
      'lily' => 'Lily Erikson'
}
```

**6. Search for the existence of a key**

```
console.log('\n6. Search for the existence of a key');
console.log('himym contains element with key lily: ' +
himym.has('lily'));
```

**Output:**

```
6. Search for the existence of a key
himym contains element with key lily
```

**7. Search for the existence of a value**

```
console.log('\n7. Search for the existence of a value');
console.log(
     'himym contains Ted Mosby: ' + [...himym.values()].includes('Ted
Mosby')
);
```

**Output:**

```
7. Search for the existence of a value
himym contains Ted Mosby
```

**8. Print keys and values using foo function**

```
console.log("\n8. Print keys and values using foo")
function foo(key) {
```

```
        console.log(key + " : " + himym.get(key))
    }
    for (const key of himym.keys()) {
        foo(key);
    }
```

**Output:**

```
8. Print keys and values using foo
marshall : Marshall Erikson
ted : Ted Mosby
barney : Barney Sitinson
lily : Lily Erikson
```

## Lua

**1. Initialize**

```
print("1. Initialize");
himym = {}
himym['marshall'] = "Marshall Erikson"
himym['ted'] = "Ted Mosby"
himym['barney'] = "Barney Stinson"
himym['robin'] = "Robin Sherbatsky"

print("Element with index marshall: " .. himym['marshall'])
print("Element with index ted     : " .. himym['ted'])
print("Element with index barney  : " .. himym['barney'])
print("Element with index robin   : " .. himym['robin'])
```

**Output:**

```
1. Initialize
Element with index marshall: Marshall Erikson
Element with index ted     : Ted Mosby
Element with index barney  : Barney Stinson
Element with index robin   : Robin Sherbatsky
```

**2. Get the value for given key**

```
print("\n2. Get the value for a given key")
print("Element with key robin: " .. himym['robin'])
```

**Output:**

```
2. Get the value for a given key
Element with key robin: Robin Sherbatsky
```

**3. Add a new element**

```
print("\n3. Add a new element")
himym['lily'] = "Lily Aldrin"
print("Element with index marshall: " .. himym['marshall'])
print("Element with index ted     : " .. himym['ted'])
print("Element with index barney  : " .. himym['barney'])
print("Element with index robin   : " .. himym['robin'])
print("Element with index lily    : " .. himym['lily'])
```

**Output:**

```
3. Add a new element
Element with index marshall: Marshall Erikson
Element with index ted     : Ted Mosby
Element with index barney  : Barney Stinson
Element with index robin   : Robin Sherbatsky
Element with index lily    : Lily Aldrin
```

**4. Remove an element**

```
print("\n4. Remove an element")
himym['robin'] = nil
print("Element with index marshall: " .. himym['marshall'])
print("Element with index ted     : " .. himym['ted'])
print("Element with index barney  : " .. himym['barney'])
print("Element with index lily    : " .. himym['lily'])
```

**Output:**

```
4. Remove an element
Element with index marshall: Marshall Erikson
Element with index ted     : Ted Mosby
Element with index barney  : Barney Stinson
Element with index lily    : Lily Aldrin
```

**5. Modify the value of an existing element**

```
 print('\n5. Modify the value of an existing element')
himym['lily'] = 'Lily Erikson'
print("Element with index marshall: " .. himym['marshall'])
print("Element with index ted     : " .. himym['ted'])
print("Element with index barney  : " .. himym['barney'])
print("Element with index lily    : " .. himym['lily'])
```

**Output:**

```
5. Modify the value of an existing element
Element with index marshall: Marshall Erikson
Element with index ted     : Ted Mosby
Element with index barney  : Barney Stinson
Element with index lily    : Lily Erikson
```

**6. Search for the existence of a key**

```
print("\n6. Search for the existence of a key")
if (himym['lily']) then
      print("himym contains element with key lily")
end
```

**Output:**

```
6. Search for the existence of a key
himym contains element with key lily
```

**7. Search for the existence of a value**

```
print("\n7. Search for the existence of a value")
for key,value in pairs(himym) do
      if (himym[key] == 'Ted Mosby') then
```

```
            print("himym contains Ted Mosby")
      end
end
```

**Output:**

```
7. Search for the existence of a value
himym contains Ted Mosby
```

**8. Print keys and values using foo function**

```
 print("\n8. Print keys and values using foo")

function foo(key, value)
      print(key .. " : " .. value)
end

for key,value in pairs(himym) do
      foo(key, himym[key])
end
```

**Output:**

```
8. Print keys and values using foo
marshall : Marshall Erikson
ted : Ted Mosby
barney : Barney Sitinson
lily : Lily Erikson
```

## PHP

**1. Initialize**

```
echo "1. Initialize";
$himym = array("marshall"=>"Marshall Erikson", "ted"=>"Ted Mosby",
      "barney" =>"Barney Stinson", "robin"=>"Robin Sherbatsky");
foreach ($himym as $key => $val) {
    echo "\n" . $key . " : " . $val;
}
```

**Output:**

```
1. Initialize
marshall : Marshall Erikson
ted : Ted Mosby
barney : Barney Stinson
robin : Robin Sherbatsky
```

**2. Get the value for given key**

```
echo "\n\n2. Get the value for a given key";
echo "\nelement with key robin: " . $himym["robin"];
```

**Output:**

```
2. Get the value for a given key
element with key robin: Robin Sherbatsky
```

**3. Add a new element**

```
echo "\n\n3. Add a new element";
$himym["lily"] = "Lily Aldrin";
foreach ($himym as $key => $val) {
    echo "\n" . $key . " : " . $val;
}
```

**Output:**

```
3. Add a new element
marshall : Marshall Erikson
ted : Ted Mosby
barney : Barney Stinson
robin : Robin Sherbatsky
lily : Lily Aldrin
```

**4. Remove an element**

```
echo "\n\n4. Remove an element";
unset($himym["robin"]);
foreach ($himym as $key => $val) {
echo "\n" . $key . " : " . $val;
}
```

**Output:**

```
4. Remove an element
marshall : Marshall Erikson
ted : Ted Mosby
barney : Barney Stinson
lily : Lily Aldrin
```

**5. Modify the value of an existing element**

```
echo "\n\n5. Modify the value of an existing element";
$himym["lily"] = "Lily Erikson";
foreach ($himym as $key => $val) {
echo "\n" . $key . " : " . $val;
}
```

**Output:**

```
5. Modify the value of an existing element
marshall : Marshall Erikson
ted : Ted Mosby
barney : Barney Stinson
lily : Lily Erikson
```

**6. Search for the existence of a key**

```
echo "\n\n6. Search for the existence of a key";
foreach ($himym as $key => $val) {
if ($key == "lily")
    echo "\nhimym contains element with key lily";
}
```

**Output:**

```
6. Search for the existence of a key
himym contains element with key lily
```

**7. Search for the existence of a value**

```
echo "\n\n7. Search for the existence of a value";
foreach ($himym as $key => $val) {
    if ($val == "Ted Mosby")
      echo "\nhimym contains element with key lily";
}
echo "\nhimym contains Ted Mosby";
```

**Output:**

```
7. Search for the existence of a value
himym contains Ted Mosby
```

**8. Print keys and values using foo function**

```
echo "\n\n8. Print keys and values using foo";
function foo($key, $val) {
    echo "\n" . $key . " : " . $val;
}
foreach ($himym as $key => $val) {
    foo($key, $val);
}
```

**Output:**

```
8. Print keys and values using foo
marshall : Marshall Erikson
ted : Ted Mosby
barney : Barney Sitinson
lily : Lily Erikson
```

# Python

**1. Initialize**

```
print("1. Initialize")
var himym = {
    'marshall': 'Marshall Erikson',
    'ted': 'Ted Mosby',
    'barney': 'Barney Sitinson',
    'robin': 'Robin Sherbatsky',
}
print(himym)
```

**Output:**

```
1. Initialize
{'marshall': 'Marshall Erikson', 'ted': 'Ted Mosby', 'barney':
'Barney Sitinson', 'robin': 'Robin Sherbatsky'}
```

**2. Get the value for given key**

```
print("\n2. Get the value for a given key")
print("element with key robin: " + himym['robin'])
```

**Output:**

```
2. Get the value for a given key
element with key robin: Robin Sherbatsky
```

**3. Add a new element**

```
print("\n3. Add a new element")
himym['lily'] = 'Lily Aldrin'
print(himym)
```

**Output:**

```
3. Add a new element
{'marshall': 'Marshall Erikson', 'ted': 'Ted Mosby', 'barney':
'Barney Sitinson', 'robin': 'Robin Sherbatsky', 'lily': 'Lily
Aldrin'}
```

**4. Remove an element**

```
print("\n4. Remove an element")
himym.__delitem__('robin')
print(himym)
```

**Output:**

```
4. Remove an element
{'marshall': 'Marshall Erikson', 'ted': 'Ted Mosby', 'barney':
'Barney Sitinson', 'lily': 'Lily Aldrin'}
```

**5. Modify the value of an existing element**

```
print("\n5. Modify the value of an existing element")
himym['lily'] = 'Lily Erikson'
print(himym)
```

**Output:**

```
5. Modify the value of an existing element
{'marshall': 'Marshall Erikson', 'ted': 'Ted Mosby', 'barney':
'Barney Sitinson', 'lily': 'Lily Erikson'}
```

**6. Search for the existence of a key**

```
print("\n6. Search for the existence of a key")
for key in himym.keys():
  if (key == 'lily'):
     print("himym contains element with key lily")
```

**Output:**

```
6. Search for the existence of a key
himym contains element with key lily
```

**7. Search for the existence of a value**

```
print("\n7. Search for the existence of a value")
for value in himym.values():
  if (value == 'Ted Mosby'):
     print("himym contains Ted Mosby")
```

**Output:**

```
7. Search for the existence of a value
himym contains Ted Mosby
```

**8. Print keys and values using foo function**

```
print("\n8. Print keys and values using foo")
def foo(key, value):
     print(key + " : " + value)
for key in himym.keys():
     foo(key, himym[key])
```

**Output:**

```
8. Print keys and values using foo
marshall : Marshall Erikson
ted : Ted Mosby
barney : Barney Sitinson
lily : Lily Erikson
```

## Ruby

**1. Initialize**

```
puts("1. Initialize")
himym = {
      'marshall': 'Marshall Erikson',
      'ted': 'Ted Mosby',
      'barney': 'Barney Sitinson',
      'robin': 'Robin Sherbatsky',
}
puts(himym)
```

**Output:**

```
1. Initialize
{:marshall=>"Marshall Erikson", :ted=>"Ted Mosby", :barney=>"Barney
Sitinson", :robin=>"Robin Sherbatsky"}
```

**2. Get the value for given key**

```
puts("\n2. Get the value for a given key")
print("element with key robin: ")
puts "#{himym[:'robin']}"
```

**Output:**

```
2. Get the value for a given key
element with key robin: Robin Sherbatsky
```

**3. Add a new element**

```
puts("\n3. Add a new element")
himym[:'lily'] = 'Lily Aldrin'
puts(himym)
```

**Output:**

```
3. Add a new element
{:marshall=>"Marshall Erikson", :ted=>"Ted Mosby", :barney=>"Barney
Sitinson", :robin=>"Robin Sherbatsky", :lily=>"Lily Aldrin"}
```

**4. Remove an element**

```
print("\n4. Remove an element")
himym.delete(:'robin')
puts(himym)
```

**Output:**

```
4. Remove an element{:marshall=>"Marshall Erikson", :ted=>"Ted
Mosby", :barney=>"Barney Sitinson", :lily=>"Lily Aldrin"}
```

**5. Modify the value of an existing element**

```
print("\n5. Modify the value of an existing element")
himym[:'lily'] = 'Lily Erikson'
puts(himym)
```

**Output:**

```
5. Modify the value of an existing element{:marshall=>"Marshall
Erikson", :ted=>"Ted Mosby", :barney=>"Barney Sitinson", :lily=>"Lily
Erikson"}
```

**6. Search for the existence of a key**

```
puts("\n6. Search for the existence of a key")
for key in himym.keys() do
  if (key == :'lily') then
     puts("himym contains element with key lily")
  end
end
```

**Output:**

```
6. Search for the existence of a key
himym contains element with key lily
```

**7. Search for the existence of a value**

```
puts("\n7. Search for the existence of a value")
for value in himym.values() do
  if (value == 'Ted Mosby') then
     puts("himym contains Ted Mosby")
  end
end
```

**Output:**

```
7. Search for the existence of a value
himym contains Ted Mosby
```

**8. Print keys and values using foo function**

```
puts "\n8. Print keys and values using foo"
def foo(key, value)
  print key
  print " : "
  puts value
end

for key in himym.keys() do
  foo(key, himym[key])
```

```
end
```

**Output:**

```
8. Print keys and values using foo
marshall : Marshall Erikson
ted : Ted Mosby
barney : Barney Sitinson
lily : Lily Erikson
```

## Rust

**1. Initialize**

```
println!("1. Initialize");
let mut himym = HashMap::new();

himym.insert(String::from("marshall"), "Marshall Erikson");
himym.insert(String::from("ted"), "Ted Mosby");
himym.insert(String::from("barney"), "Barney Stinson");
himym.insert(String::from("robin"), "Robin Sherbatsky");

for (key, value) in &himym {
    println!("{} : {}", key, value);
}
```

**Output:**

```
1. Initialize
ted : Ted Mosby
barney : Barney Stinson
robin : Robin Sherbatsky
marshall : Marshall Erikson
```

**2. Get the value for given key**

```
println!("\n2. Get the value for a given key");
println!("element with key robin: {}" , himym["robin"]);
```

**Output:**

```
2. Get the value for a given key
element with key robin: Robin Sherbatsky
```

**3. Add a new element**

```
println!("\n3. Add a new element");
himym.insert(String::from("lily"), "Lily Aldrin");
for (key, value) in &himym {
     println!("{} : {}", key, value);
}
```

**Output:**

```
3. Add a new element
ted : Ted Mosby
barney : Barney Stinson
lily : Lily Aldrin
```

```
robin : Robin Sherbatsky
marshall : Marshall Erikson
```

**4. Remove an element**

```
 println!("\n4. Remove an element");
 himym.remove("robin");
 for (key, value) in &himym {
    println!("{} : {}", key, value);
 }
```

**Output:**

```
4. Remove an element
ted : Ted Mosby
barney : Barney Stinson
lily : Lily Aldrin
marshall : Marshall Erikson
```

**5. Modify the value of an existing element**

```
println!("\n5. Modify the value of an existing element");
himym.insert(String::from("lily"), "Lily Erikson");
for (key, value) in &himym {
    println!("{} : {}", key, value);
}
```

**Output:**

```
5. Modify the value of an existing element
ted : Ted Mosby
barney : Barney Stinson
lily : Lily Erikson
marshall : Marshall Erikson
```

**6. Search for the existence of a key**

```
println!("\n6. Search for the existence of a key");
for (key, value) in &himym {
    if key == "lily" {
        println!("himym contains element with key lily")
    }
}
```

**Output:**

```
6. Search for the existence of a key
himym contains element with key lily
```

**7. Search for the existence of a value**

```
println!("\n7. Search for the existence of a value");
for (key, value) in &himym {
    if value == &"Ted Mosby" {
        println!("himym contains Ted Mosby");
    }
}
```

**Output:**

```
7. Search for the existence of a value
```

```
himym contains Ted Mosby
```

**8. Print keys and values using foo function**

```
println!("\n8. Print keys and values using foo");
for (key, value) in &himym {
   foo(key, value.to_string());
}

fn foo(key: &String, val: String) {
    println!("{} : {}", key, val)
}
```

**Output:**

```
8. Print keys and values using foo
ted : Ted Mosby
barney : Barney Stinson
lily : Lily Erikson
marshall : Marshall Erikson
```

## Readability and Writability Evaluation

**Dart:**

Initializing, adding elements, and deleting elements of associative arrays in Dart is really convenient. The bracket syntax is basic and commonly used among other programming languages which improves writability. Printing elements using $ sign is a little confusing and not ideal. Loops are needed to look for specific elements and this also decreases writability.

**JavaScript:**

Many functionalities of maps in JavaScript are basic. Their functions are easy to use and understand. There are methods for understanding if an element exists in the map, but usage can sometimes be a little too complicated. Overall, it is a highly readable and writable language in terms of associative arrays.

**Lua:**

In Lua initialization, adding, changing and deleting operations are easy. However, the language cannot print the array by itself. Therefore, additional code is needed to check the output compared to other languages. Loops are needed to figure out if a element exists but they are easy to understand, highly readable. However, I would not rate Lua as an easily writable language.

**PHP:**

Initialization, adding, changing, and deleting operations standardly easy in PHP. The language does not print associative arrays directly but looping through arrays and printing is relatively easy. The use of $ character, =>, and the echo keyword reduces

readability. The writability of the language is really high when it comes to operations of associative arrays.

**Python:**
Initialization, addition, change and deletion are really readable and writable actions in Python. The language automatically prints the dictionary. Although there are no specific methods for checking whether an item is in the array, the loops are easy to understand, they are basically English. Therefore, I would categorize Python as most readable and writeable when it comes to associative arrays.

**Ruby:**
In terms of initialization and printing ruby is fairly easy to understand and use. However, the use of hashtags while printing elements and the use of colons while getting an element by its key really complicates the procedure. I would not categorize Ruby as a easily readable and writeable programming language in terms of associative arrays.

**Rust:**
In every part of the process, I had problems with using HashMaps of Rust. First of all, insertion is unnecessarily complicated and hard to understand. Giving the type of a string in double quotes during insertion should not be necessary. Other than that, the referencing and passing arguments to functions is complicated in rust. I would categorize Rust as the worst language in terms of readability and writability.

## My Learning Strategies

While doing this homework I have tried to learn the languages and their syntax by googling. I have used Stack Overflow and official documentations from the websites. It was hard, especially for Rust, to solve the errors that I have faced. Honestly this homework took longer than anticipated because of the need for research involved. I have jumped into writing code using online compilers and learned the syntax as I proceeded. Mostly I had to experiment what will work in terms of syntax errors. I have tried many things until I found the right syntax. Even when I did find it I lacked the understanding of why the language was built the way it was.

## Online Compilers

https://dartpad.dev/
https://www.tutorialspoint.com/execute_lua_online.php
https://www.tutorialspoint.com/execute_php_online.php
https://www.tutorialspoint.com/execute_ruby_online.php
https://www.tutorialspoint.com/compile_rust_online.php

## References


[1] https://stackoverflow.com/
[2] https://www.tutorialspoint.com/dart_programming/dart_programming_map.htm
[3] https://www.javatpoint.com/dart-function#:~:text=Dart%20function%20is%20a%20set,and%20enhances%20the%20code%20reusability.
[4] https://www.lua.org/pil/5.html
[5] https://www.lua.org/pil/3.3.html
[6] https://www.w3schools.com/php/php_arrays_associative.asp
[7] https://ruby-doc.org/core-3.1.2/Hash.html
[8] https://zetcode.com/lang/rubytutorial/arrays/
[9] https://www.includehelp.com/rust/print-only-values-of-a-hashmap.aspx