CS315
Homework 1

Ata Seren
21901575

Dart:

1. What types are legal for subscripts?

```
 var arr = [1,2,3,4,5];
 print(arr);
 print("Some elements from the array:");
 print(arr[0]);
 print(arr[1]);
 print(arr[2]); //Subscripts with integers are
legal.
```

Subscripts as integers with square brackets are accepted.

2. Are subscripting expressions in element references range checked?

```
var arr = [1,2,3,4,5];
print(arr[2]);
/*
Dart checks range and code segment below
gives syntax error since subscript is out of range.
print(arr[10]);
*/
```

It checks and gives syntax error when index is out of range.

3. When are subscript ranges bound?
   There are 2 types of arrays in Dart: fixed-size and growable.
   Fixed-size array:

```
var fixedBoundArr = new List(3);
fixedBoundArr[0] = 0;
fixedBoundArr[1] = 1;
fixedBoundArr[2] = 2;
print(fixedBoundArr[2]);
```

A new element can't be added to this array. Subscript ranges bound before run time.

add() func works for arr array we used in 1st question because its range is bound in run time and can be changed. However, range of fixed-length arrays can't be changed. Because of this, following code segment will give an error.

```
add.fixedBoundArr(5);
```

Different than the "fixed-size" arrays, Dart provides us the "growable" arrays that are similar to dynamic arrays. Their ranges can be changed at any time and new data can be allocated for new elements.

Growable array:

```
var dynArr = new List();
//print(dynArr[0]);
print(dynArr.length);
dynArr.add(3);
print(dynArr.length);
dynArr.add(4);
print(dynArr.length);
dynArr.add(5);
print(dynArr.length);
print(dynArr);
```

Subscript ranges bound at run time since they are determined when `var dynArr = new List();` line works and ranges change by `add()` function.

4. When does allocation take place?

Fixed-size arrays' allocation takes place before run time because new allocations to this type of arrays are not allowed at run time.
Growable arrays' allocation takes place at run time since it is determined at run time and also can be changed at run time.

For both Question 3 and 4, I also have comments in the code that I have done more explanation.

5. Are ragged or rectangular multidimensional arrays allowed, or both?
   Both are allowed.

```
List rec1 = [1,2,3,4];
List rec2 = [5,6,7,8];
List rec3 = [9,10,11,12];
List rectangle = [rec1, rec2, rec3];
print("Rectangle 2d array:");
print(rectangle);

List rag1 = [1,2,3];
List rag2 = [5,6,7,8];
List rag3 = [9,10];
List ragged = [rag1, rag2, rag3];
print("Ragged 2d array:");
print(ragged);
print("");
```

6. Can array objects be initialized?
   Yes, we can see it by the first code segment example which array declared and initialized at the same time.

```
var arr = [1,2,3,4,5];
```

7. Are any kind of slices supported?
   Slices are supported in following formats:

```
print(arr.sublist(1, 3));
print(arr.sublist(2));
```

   Slice with 2 integers includes start and end indices of the slice (end index not inclusive).
   Slice with single integer includes only start index of the slice.

8. Which operators are provided?
   Following operations can be conducted with following operators:

```
List arr1 = [1,2,3,4,5];
List arr2 = [6,7,8,9,10];

print("Operators for arrays are tested by some code
segments.");
//Operators for arrays are tested by following code
segments.
print(arr1 + arr2);
print(arr2 == arr1);
print(arr2 != arr1);
List arr3 = [...arr1,0,...arr2];
print(arr3);
```

PHP:

1. What types are legal for subscripts?
   In PHP, an array can be created as a plain array or as a map:

```
$keyedArray = array(
"Ahmet" => "first",
"Ata" => "second",
"Can" => "third",
"Alp" => "fourth",
"Ayda" => "fifth",
);
echo: "keyedArray: ". PHP_EOL
print_r($keyedArray);
echo PHP_EOL;
$noKeyedArray = array("zero","one", "two",
"three", "four");
echo: "noKeyedArray: ". PHP_EOL
print_r($noKeyedArray);

echo "Some elements of the noKeyedArray:".
PHP_EOL;
echo $noKeyedArray[0]. PHP_EOL;
echo $noKeyedArray[1]. PHP_EOL;
echo $noKeyedArray[2]. PHP_EOL;
echo PHP_EOL;

echo: "Some elements of the keyedArray: ".
PHP_EOL
```

```
echo $keyedArray["Ahmet"]. PHP_EOL;
echo $keyedArray["Ata"]. PHP_EOL;
echo PHP_EOL;
echo "For keyed arrays, string (and integer if
assigned as key) subscripts are legal but only
integer subscripts are legal at no-keyed
arrays.". PHP_EOL;
echo PHP_EOL;
```

Therefore, a string as a map key or an integer as array index can be used as subscript.

2. Are subscripting expressions in element references range checked?

```
/*
This gives a syntax error since PHP checks ranges
of arrays.
echo $noKeyedArray[8]. PHP_EOL;
*/
```

It checks and gives syntax error when index is out of range.

3. When are subscript ranges bound?
   Ranges of all types of arrays in PHP (keyed, no keyed, dynamic) are bound at run time because they can be changed at run time.

```
echo "Manipulation in keyedArray and
noKeyedArray:"
$keyedArray[5] = "sixth";
$noKeyedArray[5] = "six";
echo "New element added to keyedArray: ";
echo $keyedArray[5] . PHP_EOL;
echo "New element added to noKeyedArray: ";
echo $noKeyedArray[5] . PHP_EOL;
echo PHP_EOL;

echo "Dynamic array implementation:". PHP_EOL;
$dynamicArray = array();
echo "Size can be increased with every append:".
PHP_EOL;
for($i = 0; $i < 5; $i++)
{
```

```
    $dynamicArray[$i] = $i;
    echo "Size: ", count($dynamicArray). PHP_EOL;
    echo $dynamicArray[$i]. PHP_EOL;
}
echo PHP_EOL;
```

4. When does allocation take place?

   Allocation takes place at run time since there are no static arrays in PHP.

5. Are ragged or rectangular multidimensional arrays allowed, or both?
   Both are allowed.

```
$multiRecArray = array (
    array("Ahmet",16,1),
    array("Ata",19,6),
    array("Derin",21,5),
    array("Seren",20,20)
);
echo "Rectangle 2d array:". PHP_EOL;
print_r($multiRecArray);

$multiJagArray = array (
    array("Ahmet",16,1,2,3,4,5),
    array("Ata",19,6,3),
    array("Derin"),
    array("Seren",20)
);
echo "Ragged 2d array:". PHP_EOL;
print_r($multiJagArray);
```

6. Can array objects be initialized?
   Yes, we can see it by the first code segment of keyed and no keyed array examples which array declared and initialized at the same time.

7. Are any kind of slices supported?
   Slices are supported in following formats:
   `print_r(array_slice($noKeyedArray,2));`

```
print_r(array_slice($noKeyedArray,-3));
print_r(array_slice($noKeyedArray,1,2));
print_r(array_slice($noKeyedArray,-2,2));
print_r(array_slice($noKeyedArray,2,2,true));
print_r(array_slice($noKeyedArray,2,2,false));
```

Negative integers indicate the array element at the end of the array. For example, -1 is the index of last element and -2 is the index of second last element.
Slice with 2 integers indicate start and end indices of the slice (end index not inclusive).
Slice with single integer indicates only start index of the slice.
Slice() in PHP returns an array and boolean value indicates whether the elements of the slice will be printed with their original indices or new indices of the created array.

8. Which operators are provided?
   Following operations can be conducted with following operators:

```
print_r( $keyedArray+$noKeyedArray);
echo PHP_EOL;
print_r( $keyedArray==$noKeyedArray);
echo PHP_EOL;
print_r( $keyedArray===$noKeyedArray);
echo PHP_EOL;
print_r( $keyedArray!=$noKeyedArray);
echo PHP_EOL;
print_r( $keyedArray<>$noKeyedArray);
echo PHP_EOL;
```

JavaScript:

(Only script part is mentioned in the report but HTML codes to display the data are available in the .html file.)

1. What types are legal for subscripts?
   Both string and integer can be used as subscripts.

```
arr = ["zero", "one", "two", "three"];
arr["someNum"] = "number";
document.getElementById("P1").innerHTML = arr;
document.getElementById("P2").innerHTML = arr[2];
document.getElementById("P3").innerHTML =
arr["someNum"];
```

   However, using a string as an index creates an element in an array with an undefined location. Therefore, when the array is printed, the element with string subscript is not included on the browser.

2. Are subscripting expressions in element references range checked?

```
/*
JS checks bound but doesn't give syntax error.
Instead, it prints "undefined".
*/
//document.getElementById("P2").innerHTML =
arr[9];
```

3. When are subscript ranges bound?
   Ranges of arrays in JavaScript are bound at run time because they can be changed at run time.

```
arr2 = [];
document.getElementById("A1").innerHTML =
arr2.length;
arr2[0] = "first";
arr2[1] = "second";
arr2[2] = 3;
document.getElementById("A2").innerHTML = arr2;
```

```
document.getElementById("A3").innerHTML =
arr2.length;
```

4. When does allocation take place?

   Allocation takes place at run time since there are no static arrays in JavaScript.

5. Are ragged or rectangular multidimensional arrays allowed, or both?
   Both are allowed.

```
arr3 = [
  ["A", 1, 2],
  ["B", 3, 4],
  ["C", 5, 6],
  ["D", 7, 8],
];
document.getElementById("B1").innerHTML = arr3;
arr4 = [
  ["A", 1, 2, 3, 4],
  ["B"],
  ["C", 5, 6],
  ["D", 7],
];
```

6. Can array objects be initialized?
   Yes, we can see it by the first code segment example which array declared and initialized at the same time.

7. Are any kind of slices supported?
   Slices are supported in following formats:

```
arr5 = ["a", "b", "c", "d", "e"];
arr5sliced1 = arr5.slice(1, 3);
arr5sliced2 = arr5.slice(-3, -1);

document.getElementById("C1").innerHTML          =
arr5sliced1;
```

```
document.getElementById("C2").innerHTML =
arr5sliced2;
```

Negative integers indicate the array element at the end of the array. For example, -1 is the index of last element and -2 is the index of second last element.

Slice with 2 integers indicate start and end indices of the slice (end index not inclusive).

8. Which operators are provided?
   Following operations can be conducted with following operators:

```
document.getElementById("D1").innerHTML = arr +
arr2;
document.getElementById("D2").innerHTML = arr ==
arr2;
document.getElementById("D3").innerHTML = arr !=
arr2;
document.getElementById("D4").innerHTML = arr <
arr;
document.getElementById("D5").innerHTML = arr <=
arr;
document.getElementById("D6").innerHTML = arr >
arr;
document.getElementById("D7").innerHTML = arr >=
arr;
```

Python:

1. What types are legal for subscripts?
   In Python, arrays may have different types of elements but subscripts use integers (or negative integers).

```
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print("Some elements from array:")
print(arr[0]) # Integers are legal in subscripts.
print(arr[1])
print(arr[-1]) # Negative integers are allowed
too.
# They start from the right side of the array.
```

2. Are subscripting expressions in element references range checked?

```
"""
Python checks the ranges of an array and gives
syntax error. Following code segment gives such
syntax error since the index in it is out of
range.
print(arr[72])
"""
```

It checks and gives syntax error when index is out of range.

3. When are subscript ranges bound?
   Ranges of arrays in Python are bound at run time because they can be changed at run time.

```
print("Size of arr: "+ str(len(arr))) # Size of
array is 5 at the beginning.
print(arr)

# We can append a new element to our array.
print("Int 6 appended")
arr = np.append(arr, 6)
print(arr)
print("Size of arr: "+ str(len(arr)))

# We can remove an element from our array too.
arr =np.delete(arr, 5)
print("Int 6 removed")
print(arr)
print("Size of arr: "+ str(len(arr)))
print()
```

4. When does allocation take place?

   Allocation takes place at run time since arrays are resizable in Python.

5. Are ragged or rectangular multidimensional arrays allowed, or both?
   Both are allowed.

```
print("Rectangle 2d array:")
arrRec = np.array([[1, 2, 3], [4, 5, 6], [7, 8,
9]]),
print(arrRec)
print()
print("Ragged 2d array:")
arrRag = np.array([[1, 2, 3, 4, 5], [6, 7, 8], [9,
10]], dtype=object )
print(arrRag)
print()
```

6. Can array objects be initialized?
   Yes, we can see it by the first code segment example or the following example which array declared and initialized at the same time.

7. Are any kind of slices supported?
   Slices are supported in following formats:

```
arrSl1 = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print("Several slices of array: ")
print(arrSl1)
print()
print(arrSl1[2:7])
print(arrSl1[5:])
print(arrSl1[:5])
print(arrSl1[-5:-2])
print(arrSl1[2:8:2])

arrSl2 = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9,
10]])
print("Several slices of array: ")
print(arrSl2)
print()
print(arrSl2[1, 1:4])
print(arrSl2[0:2, 1:4])
```

Slice with 2 integers indicate start and end indices of the slice (end index not inclusive).
In the slice with 3 integers, first 2 integers indicate start and end indices of the slice (end index not inclusive). The 3rd integer indicates that elements of the slice are grouped and amount of elements in these groups is determined according to this 3rd integer. 1st element of each group is taken to the slice and other are ignored.

8. Which operators are provided?
   Following operations can be conducted with following operators:

```
arr2 = np.array([1, 2, 3])
arr3 = np.array([4, 5, 6])
print("arr2 and arr3:")
print(arr2)
print(arr3)
print("+ operator")
print("arr2 + arr3")
print(arr2 + arr3)
print()

print("in operator")
print("2 in arr2:")
print(2 in arr2) # true
print("53 in arr2:")
print(53 in arr2) # false
print()

print("= operator") # assigns the elements of an
array to another
arr4 = arr3
print("arr4 = arr3")
print("arr4 is ")
print(arr4)
print()

print("== operator") # returns true only if the
length and elements of the arrays are equal
print("arr2 == arr3:")
print(arr2 == arr3) # false
print("arr3 == arr4:")
```

```
print(arr3 == arr4) # true
```

Rust:

1. What types are legal for subscripts?
   In Rust, array subscripts use integers.

```
let arr  = [1, 2, 3, 4, 5];
println!("{:?}",arr);
println!("arr[0] is {}",arr[0]); //Subscripts
with integers are legal.
println!("Size of arr:{}",arr.len());
```

2. Are subscripting expressions in element references range checked?

```
/*
Rust checks range and code segment below gives
a syntax error since index is out of range.
println!("arr[out of range] is {}",arr[11]);
*/
```

   It checks and gives syntax error when index is out of range.

3. When are subscript ranges bound?
   If array is static, ranges are bound at compile time:

```
println!("A different type of array called
\"mutable\".
\"mut\" keyword allows user to not add or remove
but change the elements of the array. Therefore,
range bindings and allocation of arrays with
 and without mutability take place before run
 time");

 let mut arr2  = [6, 7, 8, 9, 10];
 rr2[0] = 99;

 println!("{:?}",arr2);
```

```
println!("arr2[0] is {}",arr2[0]);
println!("\"vec!\" macro, on the other hand,
allows user to increase or decrease the size of
array. Therefore, range bindings and allocation
of arrays with vec! macro takes place at run
time.");

let mut arr3 = vec![];
println!("Size of arr2:{}",arr3.len());
arr3.push(11);
arr3.push(12);
arr3.push(13);

println!("{:?}",arr3);
println!("Size of arr3:{}",arr3.len());
println!();
```

4. When does allocation take place?

   Allocation takes place before run time if array is static.
   Allocation takes place at run time if `vec!` macro is used at declaration.

5. Are ragged or rectangular multidimensional arrays allowed, or both?
   Only rectangular multidimensional arrays are allowed.

```
println!("Rectangle multidimensional array
implementation and printing:");
let mut arr4 = [[0u8; 4]; 6];
arr4[0][1] = 33 ;
arr4[0][2] = 34 ;
arr4[0][3] = 35 ;

println!("{:?}",arr4);
 println!();
 println!("Ragged multidimensional arrays are NOT
 supported by Rust.");
 println!();
```

6. Can array objects be initialized?

Yes, we can see it by the first code segment example which array declared and initialized at the same time.

7. Are any kind of slices supported?
   Slices are supported in following formats:

```
let slice = &arr[0..2];
println!("slice is {:?}", slice);
```

   Slice with 2 integers indicate start and end indices of the slice (end index not inclusive).

8. Which operators are provided?

   No operators are provided to operate on arrays.

Which language is the best for array operations?

In my opinion, Python is the best language for array operations among the languages I gave examples above. First of all, I believe that resizable arrays of Python are very useful. They are easy to write thanks to their syntax and they allow different types of elements in them. Therefore, it makes very easy to write a program that uses related data but with different types. Also, I think that changing the size of array is beneficial. It is obvious that there is a trade-off for performance. However, its advantages are more than disadvantages. Thanks to resizable arrays, debugging is easier since there won't be errors caused by array sizes. Also, all types of arrays are under the same place. User doesn't need to remember all types and their syntax to use them efficiently in the code. PHP arrays has the same functionalities and it takes the 2nd best language for arrays for me. However, it has more types and more complicated syntax. Also, the most significant difference between Python and PHP arrays is the option to use keys. However, Python gives us the opportunity to create a map. Multi-dimensional arrays are supported too. Therefore, users don't have to struggle to have the functionalities of a multi-dimensional array. Simple syntax also provides us a better slicing and easier operations with unique operators. To sum up, Python is

the best language for array operations among the languages because its properties provide us enormous amount of readability, writability and reliability.

My learning strategy for this homework:

From our project and my previous experiences about learning the code, I know that a documentation of a language includes literally everything about that language. Therefore, I thought that I can find the information about the arrays in these documentations. I go language by language and started to check the documentation of Dart language, then PHP, JavaScript, Python and Rust, respectively.

That's when I encountered the first problem. Arrays are not called as "arrays" in every language. Therefore, it sometimes made hard to find what I was looking for. However, by checking other platforms such as W3Schools, Geeksforgeeks, etc. I found which definitions are used for arrays in different languages. Some of the documentations provide some code examples about arrays too and they helped me to answer the first question. However, I didn't just refer to the documentation and used VS Code, my favorite text editor, to create a file and run it on Dijkstra machine. I experimented on them by referring to the example codes and got the expected results for first question. For the second question, I simply manipulated the subscript and tested to see if ranges are checked.

Third and fourth questions were the hardest ones for me because I couldn't figure out when do the range bindings and allocations take place since I usually didn't question it while I was writing my code. First, I checked the documentations but couldn't find such information about time. However, I found different array types that are provided by the languages and I made the deduction that if an array type allows adding a new element to an array, its ranges and memory allocation take place at run time and if it does not allow that, take place before run time. To prove my deduction, I tested each array type at each language by trying to manipulate their data and I got the expected results according to my deductions.

For the fifth question, documents didn't help me a lot so I searched example codes of multidimensional arrays of languages on W3Schools,

Geeksforgeeks and StackOverflow. When and if I find an example, I tested it on Dijkstra machine and be sure that example is correct.

For the sixth question, I simply used the documentation and my experiments and easily answered it for each language.

For the seventh question, I checked the documentation again but also checked the code examples on the previous platforms to see how slices work. I tested the slicing methods on my code too.

Finally, on the eight and last question, I mostly conducted experiments with arrays and operators to see if the operators can be used for operations with arrays.

To summarize my learning strategy, first I checked the documentations of the languages to have a general idea about the syntax and functionalities of the languages. The example codes helped me a lot because they are official so they are reliable and they were very easy to understand since they are implemented for everyone that accesses to the document. However, these examples were not provided by every documentation so I checked the coding platforms that I always use for my coding for academic purposes and personal interests. By using the information I got from documentations and the syntax and implementation methods from code examples, I wrote and tested my code to answer the questions.

REFERENCES:

[1] https://dart.dev/guides/language/language-tour#lists

[2] https://pub.dev/documentation/piecemeal/latest/piecemeal/Array2D-class.html

[3] https://www.php.net/manual/en/language.types.array.php

[4] https://www.w3schools.com/php/php_arrays_multidimensional.asp

[5] https://www.w3schools.com/js/js_arrays.asp

[6] https://www.geeksforgeeks.org/multidimensional-array-in-javascript/

[7] https://www.w3schools.com/python/python_arrays.asp

[8] https://www.w3schools.com/python/numpy/numpy_creating_arrays.asp

[9] https://www.w3schools.com/python/numpy/numpy_array_slicing.asp

[10] https://doc.rust-lang.org/std/primitive.array.html

[11] https://www.geeksforgeeks.org/rust-array/

[12] https://doc.rust-lang.org/reference/types/array.html

[13] https://stackoverflow.com/questions/34684261/how-to-set-a-rust-array-length-dynamically