CS 315
Homework 1

Erhan Er

21801809

Section 3

Dart

1. How are the boolean values represented?

Boolean values are only represented as true or false. Other types are not accepted and compiler will give an error [2].

Example

```
//bool boolean1 = 1;      Compiler does not compile
//bool boolean2 = "ab";  Compiler does not compile
bool boolean3 = true;
bool boolean4 = false;
```

2. What operators are short-circuited?

“&&” and “||” are short circuited [1].

3. How are the results of short-circuited operators computed?

“&&” looks for “false” in the left-hand side of itself. If the left-hand side is “false”, the operator does not consider the right-hand side because the result is already decided as “false”. “||” looks for “true” in the left-hand side of itself. If the left-hand side is “true”, the operator does not consider the right-hand side because the result is already decided as “true” [1].

Example

```
boolFunction1() {
   print( "boolFunction1 returns true");
   return true;
}
boolFunction2() {
   print( "boolFunction2 return false");
   return false;
}
// It only prints false
print(false && boolFunction1());
print( "----------");
// It calls boolFunction1 and prints the result
print(true && boolFunction1());
print( "----------");
// It only prints false
print( true || boolFunction2());
print( "----------");
// It calls boolFunction2 and prints the result
print( false || boolFunction2());
```

Output

```
false
----------
boolFunction1 returns true
true
----------
```

```
true
-----------
boolFunction2 return false
false
```

4. What are the advantages about short-circuit evaluation?

It can decrease the running time especially for long functions or loops.

Example

```
time1() {
  var a = 0;
  for ( var i = 0; i < 100000000; i++ ) {
    a = a + i;
  }
  return false;
}

var stopwatch = Stopwatch()..start();
if ( time1() || time1() || time1() || time1() || time1() ||
     time1() || time1() || time1() || time1() || true ) {
   print( "without short circuit");
}
print( "Elapsed time: ${stopwatch.elapsed}");

var stopwatch2 = Stopwatch()..start();
if ( true || time1() || time1() || time1() || time1() ||
     time1() || time1() || time1() || time1() || time1() ) {
   print( "with short circuit");
}
print( "Elapsed time: ${stopwatch2.elapsed}");
```

Output

```
without short circuit
Elapsed time: 0:00:07.178000
with short circuit
Elapsed time: 0:00:00.000008
```

5. What are the potential problems about short-circuit evaluation?

It can prevent an important value from changing its content. Therefore, it can cause problems in the continuity of the program.

Example

```
var value = 0;
helper3() {
   value = 10;
   print( "helper3 assigns value to 10 and returns true");
   return true;
}
```

```
if (true || helper3()) {
   var a = value * 2;
   print( "a: $a");
}
```

Output

```
a: 0
```

JavaScript

1. How are the boolean values represented?

Everything can represent a boolean value. Empty string, null, 0, -0, undefined and uninitialized variables represent false values. All other values represent true. “!!()” and “Boolean()” functions transform a non-boolean value to its boolean representation [5].

Example

```
var object1;
var object2 = {lecture: "cs315"};
var boolean1 = Boolean( "abc");
var boolean2 = !!( "");
var boolean3 = Boolean(3);
var boolean4 = Boolean(-1);
var boolean5 = Boolean(0);
var boolean6 = !!(-0);
var boolean7 = !!(undefined);
var boolean8 = Boolean(null);
var boolean9 = !!([]);
var boolean10 = !!([1, 2, 3]);
var boolean11 = !!(object1);
var boolean12 = !!(object2);
var boolean13 = true;
var boolean14 = false;
console.log( boolean1 + " " + boolean2 + " "
     + boolean3 + " " + boolean4 + " "
     + boolean5 + " " + boolean6 + " "
     + boolean7 + " " + boolean8 + " "
     + boolean9 + " " + boolean10 + " "
     + boolean11 + " " + boolean12 + " "
     + boolean13 + " " + boolean14);
```

Output

```
true false true true false false false false true true false true
true false
```

2. What operators are short-circuited?

“&&”, “||”, “??” are short circuited [6], [7], [8].

3. How are the results of short-circuited operators computed?

“&&” looks for “false” in the left-hand side of itself. If the left-hand side is “false”, the operator does not consider the right-hand side because the result is already decided as “false”. “||” looks for “true” in the left-hand side of itself. If the left-hand side is “true”, the operator does not consider the right-hand side because the result is already decided as “true”. “??” looks for “undefined” or “null” in the left-hand side of itself. If the left-hand side is “undefined” or “null”, the operator does not consider right-hand side because the result is already decided as “true” [6], [7], [8].

Example

```
function booleanFunction1() {
     console.log( "booleanFunction1 returns true");
     return true;
}
function booleanFunction2() {
     console.log( "booleanFunction2 returns false");
     return false;
}

function booleanFunction3() {
     console.log( "booleanFunction3 returns undefined");
     return undefined;
}
var human = {
     weight: "60",
     height: "180"
};
// Calls booleanFunction1 and returns true
console.log(true && booleanFunction1());
// Does not call booleanFunction1 and returns false
console.log(false && booleanFunction1());
// Does not call booleanFunction2 and returns true
console.log(true || booleanFunction2());
// Calls booleanFunction2 and returns false
console.log(false || booleanFunction2());
/** Does not call booleanFunction3
  * Because left hand size already has a return
  * different than null of undefined. */
console.log( booleanFunction1() ?? booleanFunction3() );
/** Does call booleanFunction1
  * Because left hand side has undefined return. */
console.log( booleanFunction3() ?? booleanFunction1() );
```

Output

```
booleanFunction1 returns true
true
false
true
booleanFunction2 returns false
false
booleanFunction1 returns true
true
booleanFunction3 returns undefined
booleanFunction1 returns true
true
```

4. What are the advantages about short-circuit evaluation?

It can decrease the running time especially for long functions or loops.

Example

```
function timer() {
	var a = 0;
	for ( let i = 0; i < 10000000; i++ )
		a += i;
	return false;
}

var startTime = performance.now();
if ( timer() || timer() || timer() || timer() || timer() ||
timer() || timer() || timer() || timer() || true ) {
	console.log( "without short circuit");
}

var endTime = performance.now();
console.log( "Elapsed time: " + (endTime - startTime));

var startTime2 = performance.now();
if ( true || timer() || timer() || timer() || timer() ||
timer() || timer() || timer() || timer() || timer() ) {
	console.log( "with short circuit");
}

var endTime2 = performance.now();
console.log( "Elapsed time: " + (endTime2 - startTime2));
```

Output

```
without short circuit
Elapsed time: 791
with short circuit
Elapsed time: 1
```

5. What are the potential problems about short-circuit evaluation?

It can prevent an important computation from happening. In the example given below, it prevents helper4 from assigning “5” to global “value”. In the if statement, “value * 2” is assigned to variable “a”. However, “value” is “undefined”. Hence, “value * 2” becomes NaN. Therefor, “a” could not get a value.

Example

```
var value;
function helper4() {
	value = 5;
	console.log( "helper4 assigns value to 5 and returns
true");
	return true;
}
```

```
if (true || helper4()) {
     var a = value * 2;
     console.log( "a: " + a);
}
```

Output

```
a: NaN
```

PHP

1. How are the boolean values represented?

0, “0”, empty arrays, empty strings, 0.0, -0.0, NULL and “false” are represented as false [12].

Example

```
var_dump((bool) true);       //bool(true)
var_dump((bool) false);      //bool(false)
var_dump((bool) 0);          //bool(false)
var_dump((bool) "0");        //bool(false)
var_dump((bool) "");         //bool(false)
var_dump((bool) "cs315");    //bool(true)
var_dump((bool) 315);        //bool(true)
var_dump((bool) -1);         //bool(true)
var_dump((bool) []);         //bool(false)
var_dump((bool) [1, 2, 3]); //bool(true)
var_dump((bool) NULL);       //bool(false)
var_dump((bool) 0.0);        //bool(false)
var_dump((bool) -0.0);       //bool(false)
var_dump((bool) 1.5);        //bool(true)
```

2. What operators are short-circuited?

“&&”, “and”, “||” and “or” operators are short circuited [11].

3. How are the results of short-circuited operators computed?

“&&” and “and” will check its left-hand side first. If the left-hand side is false, then the rest of the operation will not be checked because the result is already decided as false. “||” and “or” will check its left-hand side first. If the left-hand side is true, then the rest of the operation will not be checked because the result is already decided as true [11].

Example

```
function helper1() {
    echo "helper1 returns true \n";
    return true;
}

/* It will not call helper1.
 * Because of false,
 * result is already decided.
 **/
var_dump(false && helper1());
print "---------- \n";

/* It will call helper1.
 * Because of true,
 * it needs to check the other side as well.
 **/
var_dump(true && helper1());
print "---------- \n";
```

```
/* It will not call helper1.
 * Because of false,
 * result is already decided.
 **/
var_dump(false and helper1());
print "---------- \n";

/* It will call helper1.
 * Because of true,
 * it needs to check the other side as well.
 **/
var_dump(true and helper1());
print "---------- \n";

/* It will not call helper1.
 * Because of true,
 * result is already decided.
 **/
var_dump(true || helper1());
print "---------- \n";

/* It will call helper1.
 * Because of false,
 * it needs to check the other side as well.
 **/
var_dump(false || helper1());
print "---------- \n";

/* It will not call helper1.
 * Because of true,
 * result is already decided.
 **/
var_dump(true or helper1());
print "---------- \n";

/* It will call helper1.
 * Because of false
 * it needs to check the other side as well.
 **/
var_dump(false or helper1());
print "---------- \n";
```

Output

```
bool(false)
----------
helper1 returns true
bool(true)
----------
bool(false)
----------
```

```
helper1 returns true
bool(true)
----------
bool(true)
----------
helper1 returns true
bool(true)
----------
bool(true)
----------
helper1 returns true
bool(true)
----------
```

4. What are the advantages about short-circuit evaluation?

It can decrease the running time especially for long functions or loops.

Example

```
function timer() {
    $a = 0;
    for ($i = 0; $i < 10000000; $i++) {
        $a = $a + 1;
    }
    return false;
}

$startTime = microtime(true);
if ( timer() || timer() || timer() || timer() || timer() ||
timer() || timer() || timer() || timer() || true ) {
    echo "without short circuit \n";
}
$endTime = microtime(true);
print "Elapsed time: ";
var_dump($endTime - $startTime);

$startTime = microtime(true);
if ( true || timer() || timer() || timer() || timer() ||
timer() || timer() || timer() || timer() || timer() ) {
    echo "with short circuit \n";
}
$endTime = microtime(true);
print "Elapsed time: ";
var_dump($endTime - $startTime);
```

Output

```
without short circuit
Elapsed time: float(0.6500070095062256)
with short circuit
Elapsed time: float(9.5367431640625E-7)
```

5. What are the potential problems about short-circuit evaluation?

It can prevent a program from ending in a menu or a loop. In the given example below, it prevents the value “a” from assigning it to 10. Therefore, the program cannot be exited.

Example

```
$value = NULL;
function helper2() {
    global $value;
    $value = 5;
    echo "helper2 assigns value to 5 and returns true \n";
    return true;
}

if ( true || helper2() ) {
    $a = $value * 2;
    if ( $a == 10 )
        exit;
// Because a is not assigned to 10 from value * 2
// The program will not stop if this was in a menu or a loop.
}
```

Python

1. How are the boolean values represented?

Empty strings, empty lists, empty tulips, false keyword and None represent false. In addition, objects with "__len__" function which are returning 0 or false represent false. Other things represent true. "bool()" function transform a data which is not boolean to boolean representation of it [16].

Example

```
class object1:
     def __len__(self):
          return 0

class object2:
     def __len__(self):
          return 5

class object3:
     def __len__(self):
          return True

class object4:
     def __len__(self):
          return False

newObject1 = object1()
newObject2 = object2()
newObject3 = object3()
newObject4 = object4()

print( True)
print( False)
print( bool(5))
print( bool(0))
print( bool(""))
print( bool("cs315"))
print( bool([]))
print( bool(["cs", "315"]))
print( bool({}))
print( bool({3, 1, 5}))
print( bool(None))

print( bool(newObject1))
print( bool(newObject2))
print( bool(newObject3))
print( bool(newObject4))
```

Output

```
True
```

```
False
True
False
False
True
False
True
False
True
False
False
True
True
False
```

2. What operators are short-circuited?

“and”, “or”, “<”, “<=”, “>” and “>=” are short circuited [15].

3. How are the results of short-circuited operators computed?

“&&”, “and”, “<”, “<=”, “>”, “>=” and “==” will check its left-hand side first. If the left-hand side is false, then the rest of the operation will not be checked because the result is already decided as false. “||” and “or” will check its left-hand side first. If the left-hand side is true, then the rest of the operation will not be checked because the result is already decided as true [15].

Example

```
def helper():
     print("Helper function returns true")
     return True

# It does not call helper
# because of false
# the result is already found
print( False and helper())
print( "----------")

# It calls helper
# becasue of true
# it also needs to check the other side of and
print( True and helper())
print( "----------")

# It does not call helper
# because of true
# the result is already found
print( True or helper())
print( "----------")

# It calls helper
# because of false
# it also needs to check the other side
print( False or helper())
print( "----------")

def helper2(i):
     print( "helper2 returned ", i)
     return i

# When <, <=, >=, >, == see false in the first comparison
# it will stop the execution of the rest
print( 10 > 15 > helper2(5))
print( "----------")

print( 10 >= 15 >= helper2(5))
print( "----------")

print( 15 < 10 > helper2(5))
print( "----------")

print( 15 <= 10 <= helper2(5))
print( "----------")

print( 10 != 10 != helper2(5))
print( "----------")

print( 10 == 5 == helper2(5))
print( "----------")
```

Output

```
False
----------
Helper function returns true
True
----------
True
----------
Helper function returns true
True
----------
False
----------
False
----------
False
----------
False
----------
False
----------
```

```
False
----------
```

4. What are the advantages about short-circuit evaluation?

It can decrease the running time especially for long functions or loops.

Example

```
def timer():
    a = 0
    for i in range(0, 1000000):
        a = a + 1;
    return False
start = time.time()
if ( timer() or timer() or timer() or timer() or timer() or
timer() or timer() or timer() or True ):
    print( "without short circuit")
end = time.time()
print( "Elapsed Time: ", end - start)

start = time.time()
if ( True or timer() or timer() or timer() or timer() or
timer() or timer() or timer() or timer() ):
    print( "with short circuit")
end = time.time()
print( "Elapsed Time: ", end - start)
```

Output

```
without short circuit
Elapsed Time:  0.4252943992614746
with short circuit
Elapsed Time:  3.5762786865234375e-06
```

5. What are the potential problems about short-circuit evaluation?

It can prevent an important computation from happening. In the example given below, it prevents helper3 from assigning “5” to global “value”. In the if statement, “value * 5” is assigned to variable “a”. However, “value” is “None”. Hence, “value * 5” cannot be computed. Hence, “a” could not get a value.

Example

```
value = None
def helper3():
    global value
    value = 5
    print( "helper3 assigns value to 5 and returns true")
    return True

if (True or helper3()):
    # This will give error
    # Because of true,
    # “or” do not call helper3
    # this means “none * 5” will be assigned to “a”
    # but “none * 5” is not a defined operation in python.
    # Therefore, it will break the program.
    #a = value * 5
    #print( "a: ", a)
     print( “end”)
```

Output

```
end
```

Rust

1. How are the boolean values represented?

Boolean types can only be assigned as “True” or “False”. Other types cannot be converted into boolean [19].

Example

```
// Boolean types only be assigned to true or false
// other data types are not allowed.
//let bool1: bool = 1;
//let bool2: bool = "abc";
let bool3: bool = true;
let bool4: bool = false;

println!( "bool3: {}", bool3);
println!( "bool4: {}", bool4);
```

Output

```
bool3: true
bool4: false
```

2. What operators are short-circuited?

“&&” and “||” are short circuited [20].

3. How are the results of short-circuited operators computed?

“&&” operator looks its left-hand side. If left-hand side is false, then the whole operation is false. Therefore, it does not check the right-hand side. “||” operator looks its left-hand side. If the left-hand side is true, then whole operation is true. Therefore, it does not check the right-hand side [20].

Example

```
fn helper1() -> bool {
     println!( "helper1 returns true");
     return true;
}

// It will call helper1
// Because of true, it also needs to check the other side
println!( "boolOp1: {}", true && helper1());

// It will not call helper1
// Because of false, it does not need the other side
println!( "boolOp2: {}", false && helper1());

// It will not call helper1
// Because of true, ,t does not need the other side
println!( "boolOp3: {}", true || helper1());

// It will call helper1
// Because of false, it also needs to check the other side
println!( "boolOp4: {}", false || helper1());
```

Output

```
helper1 returns true
boolOp1: true
boolOp2: false
boolOp3: true
helper1 returns true
boolOp4: true
```

4. What are the advantages about short-circuit evaluation?

It can decrease the running time especially for long functions or loops.

Example

```
fn timer() -> bool {
   for i in 0..1000000 {
      let _a = i;
   }
   return false;
}

let mut start = Instant::now();
if timer() || timer() || timer() || timer() || timer() ||
     timer() || timer() || timer() || timer() || true {
   println!( "without short circuit");
}
let mut duration = start.elapsed();
println!( "Time Elapsed: {:?}", duration);

start = Instant::now();
if true || timer() || timer() || timer() || timer() || timer()
     || timer() || timer() || timer() || timer() {
   println!( "with short circuit");
}
duration = start.elapsed();
println!( "Time Elapsed: {:?}", duration);
```

Output

```
without short circuit
Time Elapsed: 363.831347ms
with short circuit
Time Elapsed: 3.703µs
```

5. What are the potential problems about short-circuit evaluation?

It can prevent an important value from changing its content. Therefore, it can cause problems in the continuity of the program.

Example

```
static mut VALUE: i32 = 20;
fn helper2() -> bool {
   unsafe {
      VALUE = 5;
   }
   println!( "helper2 assigns value to 5 and returns true");
   return true;
}

if true || helper2() {
   unsafe {
      let a = VALUE * 2;
      println!( "a: {}", a);
   }
}
```

Output

```
a: 40
```

Best Language for Short-Circuit Evaluation

Python, JavaScript and PHP allow users to have more data types that convertible to boolean. It can be important when checking whether the element has a content inside or not. However, PHP also gives false for “0” and “0.0” even if these strings are not empty. Rust and Dart forces users to use “true” and “false” in boolean variables. Therefore, Python and JavaScript is better at boolean variable range. Even if PHP has a wide range in boolean variables, the special cases can be confusing and risky. Python has more short-circuited operators. Other than “and” and “or”, it also provides “<”, “>”, “<=”, “>=” and “==” operators. JavaScript provides “&&”, “||” and “??”. “&&” and “||” works as the same Python’s “and” and “or”. In addition to those, “??” looks for “undefined” and “null” variables. In the other hand, other languages only provide “&&”, “||” and their word representations. Therefore, Python and JavaScript have more supported operators. However, Python also has more provided operators than JavaScript. Therefore, Python is better at number of supported operators. In overall, Python is the best language for short-circuit evaluation in among these languages.

Learning Strategy

For Dart, I read the documentation for boolean data type and logical operations [1], [2]. Even if I read the documentation for boolean data type, I tried string and integer values. For logical operations, I wrote a simple function in order to call it in logical operations. In addition, I tried to show the good side of the short-circuiting by measuring the time needed to calculate a short-circuited and not short-circuited logical operations. To do that, I searched the functions that calculating the elapsed time in a code segment in “stack overflow” [4]. In the final question, I tried to come up with a side effect of the short-circuited logical operations. I used an online compiler while writing the code and then I compiled it on Dijkstra machine [3]. For JavaScript, I already know most of the variables that can be converted to boolean data type. However, for some of them I had to read the documentation [5]. For logical operations, I already know the short-circuited operators. However, in order to verify my knowledge, I read the documentation of these operators [6], [7], [8]. For the questions 4 and 5, I used the same technique that I used in Dart. In order to do that, I searched the functions that calculating the elapsed time in a code segment in “stack overflow” [9]. I compiled my code in Mozilla Firefox. For PHP, I read the documentation for boolean data type and logical operations [11], [12]. For boolean variables, almost all of them was written in the documentation. Therefore, I could not try to convert new variables into boolean. For logical operations, I wrote a simple function in order to call it in logical operations and then tried to find how the short-circuit works in PHP. For the questions 4, I used the same technique that I used in Dart. In order to do that, I searched the functions that calculating the elapsed time in a code segment in “GeeksforGeeks” [14]. For question 5, I tried to exit from the code if the short-circuiting did not have side effect. In order to do that, I searched for exit code of PHP and found it in documentation [10]. I compiled PHP code in an online compiler than tried it on Dijkstra machine [13]. For Python, I read the documentation for boolean data type and logical operations [15], [16]. For boolean variables, almost all of them was written in the documentation. Therefore, I could not try to convert new variables into boolean. For logical operations, I wrote a simple function in order to call it in logical operations. For the questions 4 and 5, I used the same technique that I used in Dart. In order to do that, I searched the functions that calculating the elapsed time in a code segment [18]. I compiled the code on an online compiler and then I compiled it on Dijkstra machine as Python3 [17]. For Rust, I did the same things that I did in Dart [19], [20], [22]. I compiled it on an online compiler and then compiled it on Dijkstra machine [21].

References

[1] “Dart Programming – Logical Operators”. https://www.tutorialspoint.com/dart_programming/dart_programming_logical_operators.htm. [Accessed: Dec 7, 2021].

[2] “Dart Programming – Data Types”. https://www.tutorialspoint.com/dart_programming/dart_programming_data_types.htm. [Accessed: Dec 7, 2021].

[3] “DartPad”. https://dartpad.dev/. [Accessed: Dec 7, 2021].

[4] “Dart is there a way to measure execution time for a small code”. https://stackoverflow.com/questions/16955157/dart-is-there-a-way-to-measure-execution-time-for-a-small-code/16955257. [Accessed: Dec 8, 2021].

[5] “Boolean”. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Boolean. [Accessed: Dec 7, 2021].

[6] “Logical AND (&&)”. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_AND. [Accessed: Dec 7, 2021].

[7] “Logical OR (||)”. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_OR. [Accessed: Dec 7, 2021].

[8] “Nullish coalescing operator (??)”. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Nullish_coalescing_operator. [Accessed: Dec 7, 2021].

[9] “How to measure time taken by a function to execute”. https://stackoverflow.com/questions/313893/how-to-measure-time-taken-by-a-function-to-execute. [Accessed: Dec 8, 2021].

[10] “exit”. https://www.php.net/manual/en/function.exit.php. [Accessed: Dec 7, 2021].

[11] “Logical Operators”. https://www.php.net/manual/en/language.operators.logical.php. [Accessed: Dec 7, 2021].

[12] “PHP Data Types”. https://www.w3schools.com/Php/php_datatypes.asp. [Accessed: Dec 7, 2021].

[13] “Online PHP Editor”. https://paiza.io/projects/UhAMUu_FjsC-g8IhIxknqA. [Accessed: Dec 7, 2021].

[14] “Measuring script execution time in PHP”. https://www.geeksforgeeks.org/measuring-script-execution-time-in-php/. [Accessed: Dec 8, 2021].

[15] “Short Circuiting Techniques in Python”. https://www.geeksforgeeks.org/short-circuiting-techniques-python/. [Accessed: Dec 7, 2021].

[16] “Python Booleans”. https://www.w3schools.com/python/python_booleans.asp. [Accessed: Dec 7, 2021].

[17] “Python Online Compiler”. https://www.programiz.com/python-programming/online-compiler/. [Accessed: Dec 7, 2021].

[18] “Calculate Time taken by a Program to Execute in Python”. https://www.studytonight.com/post/calculate-time-taken-by-a-program-to-execute-in-python. [Accessed: Dec 8, 2021].

[19] “Boolean type”. https://doc.rust-lang.org/reference/types/boolean.html. [Accessed: Dec 7, 2021].

[20] “Lazy boolean operators”. https://doc.rust-lang.org/reference/expressions/operator-expr.html#lazy-boolean-operators. [Accessed: Dec 7, 2021].

[21] “Rust Playground”. https://play.rust-lang.org/. [Accessed: Dec 7, 2021].

[22] “Duration and Calculation”. https://rust-lang-nursery.github.io/rust-cookbook/datetime/duration.html. [Accessed: Dec 8, 2021].