# CS315 – Homework Assignment 2

Burak Ozturk - Section 3 - 21901841

## General Answers

### How are the results of short-circuited operators computed? (Consider also function calls)

For && operator (usage: oper1 = oper2 && oper3), implementation needs to be something like oper1 = oper2 ? oper3 : false. Thus if oper2 is false, oper3 won't be evaluated ever.

For || operator (usage: oper1 = oper2 || oper3), implementation needs to be something like oper1 = oper2 ? true : oper3. Thus if oper2 is true, oper3 won't be evaluated ever.

For other operators (^ for this example) (usage: oper1 = oper2 ^ oper3), result is tied to both inputs for all cases, thus there can't be short-circuit evaluation.

### What are the advantages about short-circuit evaluation?

Considering a line as "x = y <operator> z;" in a language that applies short-circuit evaluation for <operator>.

1. If z is a time/energy-costly expression, by-passing it can be very good for efficiency.
2. Lets one to set y as checkpoint to check z for runtime errors without actually evaluate it. Example: if ( isInitialized(arr) && arr[3]) …

### What are the potential problems about short-circuit evaluation?

1. It may lead to unexpected behavior for unfamiliar users. For example, if z is a function, assuming it will be called every time that line evaluated would be wrong.
2. Extra checks and steps done to make the short-circuit failsafe may be more costly that the z itself for some situations.

## Dart

### Code segment:

```
bool returnTrue() {
      print("True called");
      return true;
}

void main() {
      // 1. How are the boolean values represented?
      bool b = true;
      print("True: $b");
      b = false;
      print("False: $b");

      // 2. What operators are short-circuited?

      // Bitwise AND and OR: &, |

      print("\nIf operator is short-circuited, 'True called' will get
printed less than two times\nelse operator is not short-circuited.");

      print("\n&&:");
      true && returnTrue();
      false && returnTrue();

      print("\n||:");
      true || returnTrue();
      false || returnTrue();

      print("\n^:");
      true ^ returnTrue();
      false ^ returnTrue();
}
```

### Console output:

```
True: true
False: false

If operator is short-circuited, 'True called' will get printed less than
two times else operator is not short-circuited.

&&:
True called

||:
True called

^:
True called
True called
```

### How are the boolean values represented?

True/1: true

False/0: false

## What operators are short-circuited?

Operators && and || are short-circuited.

&& (usage: oper1 && oper2) will by-pass oper2 if oper1 is false.

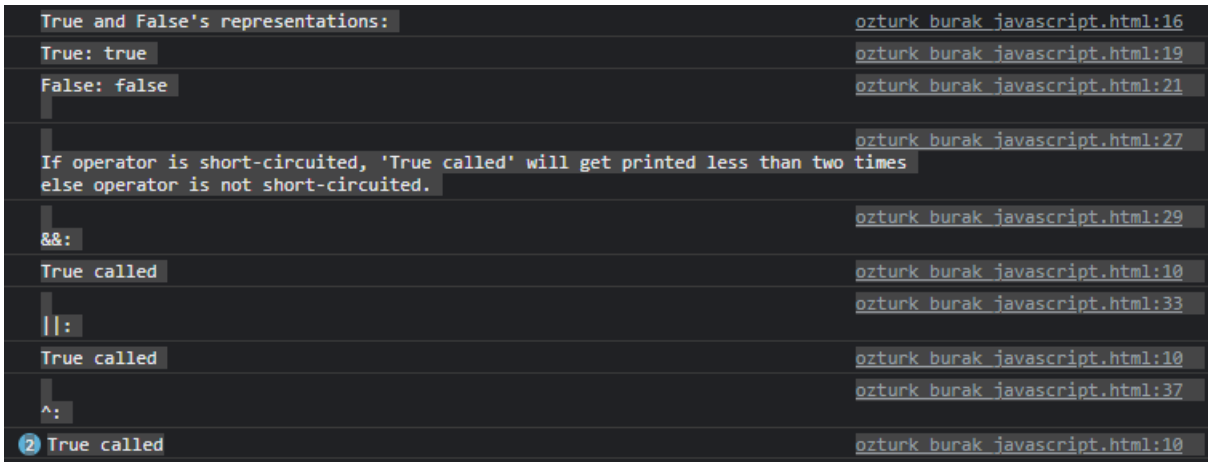
|| (usage: oper1 || oper2) will by-pass oper2 if oper1 is true.

## JavaScript

## Code Segment:

```
<html>
    <head>
        <title>Homework 2</title>
    </head>
    <body>
            <p> I used the console to do the printing </p>

            <script>
                  function returnTrue() {
                        console.log("True called\n");
                        return true;
                  }

                  // 1. How are the boolean values represented?

                  console.log("True and False's representations:\n");

                  var b = true;
                  console.log(`True: ${b}\n`);
                  b = !b;
                  console.log(`False: ${b}\n\n`);

                  // 2. What operators are short-circuited?

                  // AND and OR: &&, ||

                  console.log("\nIf operator is short-circuited, 'True
called' will get printed less than two times\nelse operator is not short-
circuited.\n");

                  console.log("\n&&:\n");
                  true && returnTrue();
                  false && returnTrue();

                  console.log("\n||:\n");
                  true || returnTrue();
                  false || returnTrue();

                  console.log("\n^:\n");
                  true ^ returnTrue();
                  false ^ returnTrue();
            </script>
      </body>
</html>
```

## Console output:




## How are the boolean values represented?

True/1: true

False/0: false

## What operators are short-circuited?

Operators && and || are short-circuited.

&& (usage: oper1 && oper2) will by-pass oper2 if oper1 is false.

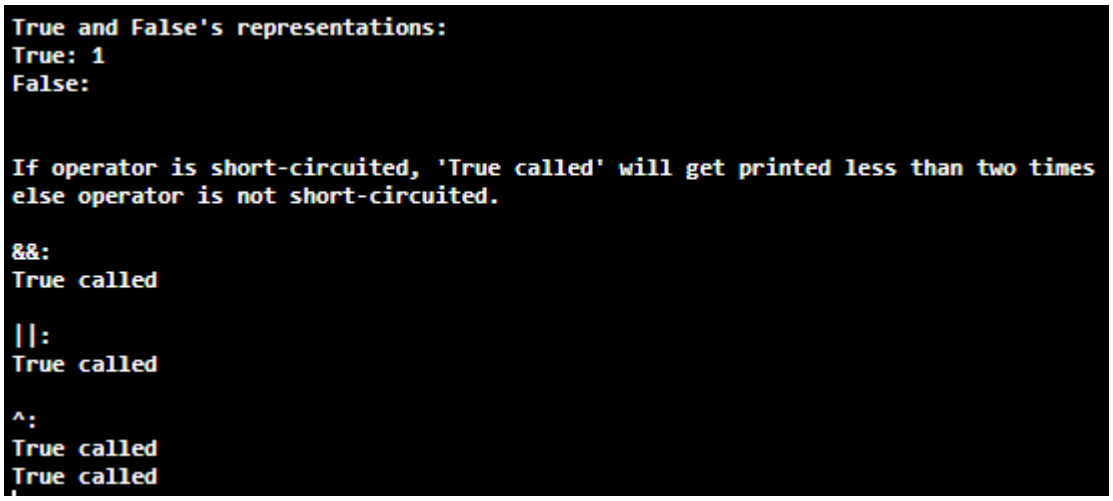
|| (usage: oper1 || oper2) will by-pass oper2 if oper1 is true.

## PHP

### Code Segment:

```
<!DOCTYPE html>
<html>
	<body>
		<?php

			function returnTrue() {
				echo "True called\n";
				return true;
			}

			// 1. How are the boolean values represented?

			echo "True and False's representations:\n";

			$b = true;
			echo "True: " . $b . "\n";
			echo "False: " . !$b . "\n\n";

			// 2. What operators are short-circuited?

			// AND and OR: &&, ||

			echo "\nIf operator is short-circuited, 'True called'
will get printed less than two times\nelse operator is not short-
circuited.\n";

			echo "\n&&:\n";
			true && returnTrue();
			false && returnTrue();

			echo "\n||:\n";
			true || returnTrue();
			false || returnTrue();

			echo "\n^:\n";
			true ^ returnTrue();
			false ^ returnTrue();
		?>
	</body>
</html>
```

## Console output:




## How are the boolean values represented?

True/1: 1

False/0:

## What operators are short-circuited?

Operators && and || are short-circuited.

&& (usage: oper1 && oper2) will by-pass oper2 if oper1 is false.

|| (usage: oper1 || oper2) will by-pass oper2 if oper1 is true.

## Python

### Code Segment:

```
def returnTrue():
    print("True called")
    return True

# 1. How are the boolean values represented?

print("True and False's representations:")

b = True;
print(f"True: {b}")
print(f"False: {not b}\n")

# 2. What operators are short-circuited?

# AND and OR: and, or

print("If operator is short-circuited, 'True called' will get printed less
than two times\nelse operator is not short-circuited.")

print("\nand:")
True and returnTrue();
False and returnTrue();

print("\nor:")
True or returnTrue();
False or returnTrue();

print("\n^:")
True ^ returnTrue();
False ^ returnTrue();
```

### Console output:

```
True and False's representations:
True: True
False: False

If operator is short-circuited, 'True called' will get printed less than
two times
else operator is not short-circuited.

and:
True called

or:
True called

^:
True called
True called
```

## How are the boolean values represented?

True/1: True

False/0: False

## What operators are short-circuited?

Operators && and || are short-circuited.

&& (usage: oper1 && oper2) will by-pass oper2 if oper1 is false.

|| (usage: oper1 || oper2) will by-pass oper2 if oper1 is true.

## Rust

### Code Segment:

```
fn return_true() -> bool {
    println!("True called");
    return true;
}

fn main() {
      // 1. How are the boolean values represented?

    let b = true;
    println!("True: {}", b);
    println!("False: {}", !b);

    // 2. What operators are short-circuited?

      // AND and OR: &&, ||

      println!("\nIf operator is short-circuited, 'True called' will get
printed less than two times\nelse operator is not short-circuited.");

      println!("\n&&:");
      true && return_true();
      false && return_true();

      println!("\n||:");
      true || return_true();
      false || return_true();

      println!("\n^:");
      true ^ return_true();
      false ^ return_true();
}
```

## Console output:

```
True: true
False: false

If operator is short-circuited, 'True called' will get printed less than two times
else operator is not short-circuited.

&&:
True called

||:
True called

^:
True called
True called
```

## How are the boolean values represented?

True/1: true

False/0: false

## What operators are short-circuited?

Operators && and || are short-circuited.

&& (usage: oper1 && oper2) will by-pass oper2 if oper1 is false.

|| (usage: oper1 || oper2) will by-pass oper2 if oper1 is true.

## Final Questions:

1. In all five of the languages, short-circuit operators were && (and/or keyword “and”) and || (and/or keyword “or”). Since there are no deviation nor difference between five languages in terms of short-circuit evaluation mechanic, I explained good and bad sides of the practices at the beginning under the title “General Answers”.

   Advantages are increased efficiency and a good way to prevent certain runtime errors. This two advantages are present for all five languages.

   Disadvantages are possible unexpected behavior and possible extra runtime cost than it saves. This disadvantages possibly exist in all five languages as well.

   In this situation I would say all five language are in a tie but since Python is much more clear and explanatory than others in general syntax and error messages, I think Python handles complexities like short-circuit evaluation very well.

2. Before Homework 1, I didn't know 4/5 of languages asked. Since I am operating in a very unknown domain for me, I am starting with learning basics of each language.

   Initial getting ready:
   Variables, functions, standard output, arrays etc. need to be learned to perform basic things that homeworks ask for. After learning very basics of each language, I search for the domain of the homework (Boolean operators' dynamics for this one). After researching, I start some sort of IDE and start experimenting. I try different things until I get decent enough to write basic scripts like Hello World, array operations, Fibonacci etc.

   Homework set-up:
   For each language and each question of the homework, I design a code snippet to prove my answer. For example, if I say JavaScript doesn't have some functionality, the code snippet corresponding that part should prove that JavaScript really doesn't have that functionality. While writing all those little snippets, languages oftenly doesn't work like I think they would. When that happens, I go back to researching and find out about that part I don't know about the language. After that I continue from where I left before.

   Writing report:
   Again for each language and each question, I add a console output and write an explanation. I am trying to write explanations clear and by staying on question domain without wandering away from the subject. After all questions are answered and supplied with examples, I answer the final questions the same way as others.

   Materials and tools used:

   IDLE Shell 3.10.0 for Python
   Opera Developer Console for JavaScript
   https://www.jdoodle.com for Dart, PHP and Rust

   Various tutorial and guiding sites for gathering info, including but not limited to;
   w3schools.com
   tutorialspoint.com
   geeksforgeeks.org
   programiz.com
   Languages' own documentations
   stackoverflow.com
   quora.com