CS 315 Homework 2

Logically Controlled Loops in Dart, Javascript, Lua, PHP, Python, Ruby, and Rust

Ant Duru

ID: 21704108 Section: 02

DART

CODE:

```
void main(){

  // pretest or posttest in while loop
  var myVar = 5;
  var pretest = true;
  print("-----------CHECK IF PRETEST OR POSTTEST FOR WHILE
LOOP-----------");

  // check if dart is posttest in while loop
  while(myVar >= 10){
    pretest = false;
    print("Dart is posttest for while loop");
  }

  // check if dart is pretest in while loop
  if(pretest == true){
    print("Dart is pretest for while lopp");
  }

  // pretest or posttest in do-while loop
  var myVar2 = 5;
  var pretest2 = true;
  print("-----------CHECK IF PRETEST OR POSTTEST FOR WHILE
LOOP-----------");
  do {
    pretest2 = false;
    print("Dart is posttest for do-while loop");
    myVar2++;
  } while (myVar2 >= 10);

  if (pretest2 == true) {
    print ("Dart is pretest for do-while loop");
  }

  // break statement for while loop
  var value = 2;
  var loopCount = 1;
  print("-----------BREAK STATEMENT FOR WHILE LOOP-----------");
  while(value <= 10){
    print("loop: $loopCount , value: $value");
```

```
  value++;
  loopCount++;

  //check if break
  if(value == 7){
    break;
  }
}
print("end of while loop");
print("loop uses break when the value is $value");

// break statement for do-while loop
var value2 = 2;
var loopCount2 = 1;
print("-----------BREAK STATEMENT FOR DO-WHILE LOOP-----------");
do{
  print("loop: $loopCount2 , value: $value2");
  value2++;
  loopCount2++;

  //check if break
  if(value2 == 5){
    break;
  }
}while(value2 <= 10);
print("end of do-while loop");
print("loop uses break when the value is $value2");

// break statement for for loop
print("-----------BREAK STATEMENT FOR FOR LOOP-----------");
var lc = 1;
for(var i = 3; i < 9; i++){
  print("loop: $lc , value: $i");
  lc++;
  if(i == 8){
    print("value is now $i , loop now uses break");
    break;
  }
}
print("end of for loop");

// continue statement for while loop
var value3 = 2;
```

```
var loopCount3 = 1;
var cValue;
print("----------CONTINUE STATEMENT FOR WHILE LOOP----------");
while(value3 <= 10){
  value3++;
    //check if continue
    if(value3 == 7){
      cValue = 7;
      loopCount3++;
      print("skipped when value is 7");
      continue;
    }
    print("loop: $loopCount3 , value: $value3");
    loopCount3++;
}
print("end of while loop");
print("loop uses continue when the value is $cValue");

// continue statement for do-while loop
var value4 = 2;
var loopCount4 = 1;
var cValue2;
print("----------CONTINUE STATEMENT FOR DO-WHILE LOOP----------");
do {
  value4++;

  //check if continue
  if (value4 == 5) {
    cValue2 = 5;
    loopCount4++;
    print("skippen when value is:  $cValue2");
    continue;
  }

  print("loop: $loopCount4 , value: $value4");
  loopCount4++;
} while (value4 <= 10);
print("end of do-while loop");
print("loop uses continue when the value is $cValue2");

// continue statement for for loop
 print("----------CONTINUE STATEMENT FOR FOR LOOP----------");
 var lc2 = 1;
```

```
    for(var i = 3; i < 9; i++){
      if(i == 5){
        print("value is now $i , loop now uses continue");
        continue;
      }
      print("loop: $lc2 , value: $i");
      lc2++;
    }
    print("end of for loop");
}
```

Output:

```
 $dart main.dart
-----------CHECK IF PRETEST OR POSTTEST FOR WHILE LOOP-----------
Dart is pretest for while lopp
-----------CHECK IF PRETEST OR POSTTEST FOR WHILE LOOP-----------
Dart is posttest for do-while loop
-----------BREAK STATEMENT FOR WHILE LOOP-----------
loop: 1 , value: 2
loop: 2 , value: 3
loop: 3 , value: 4
loop: 4 , value: 5
loop: 5 , value: 6
end of while loop
loop uses break when the value is 7
-----------BREAK STATEMENT FOR DO-WHILE LOOP-----------
loop: 1 , value: 2
loop: 2 , value: 3
loop: 3 , value: 4
end of do-while loop
loop uses break when the value is 5
-----------BREAK STATEMENT FOR FOR LOOP-----------
loop: 1 , value: 3
loop: 2 , value: 4
loop: 3 , value: 5
loop: 4 , value: 6
loop: 5 , value: 7
```

```
loop: 6 , value: 8
value is now 8 , loop now uses break
end of for loop
-----------CONTINUE STATEMENT FOR WHILE LOOP-----------
loop: 1 , value: 3
loop: 2 , value: 4
loop: 3 , value: 5
loop: 4 , value: 6
skipped when value is 7
loop: 6 , value: 8
loop: 7 , value: 9
loop: 8 , value: 10
loop: 9 , value: 11
end of while loop
loop uses continue when the value is 7
-----------CONTINUE STATEMENT FOR DO-WHILE LOOP-----------
loop: 1 , value: 3
loop: 2 , value: 4
skippen when value is:  5
loop: 4 , value: 6
loop: 5 , value: 7
loop: 6 , value: 8
loop: 7 , value: 9
loop: 8 , value: 10
loop: 9 , value: 11
end of do-while loop
loop uses continue when the value is 5
-----------CONTINUE STATEMENT FOR FOR LOOP-----------
loop: 1 , value: 3
loop: 2 , value: 4
value is now 5 , loop now uses continue
loop: 3 , value: 6
loop: 4 , value: 7
loop: 5 , value: 8
 end of for loop
```

Explanation:

There are two types of logically controlled loops in Dart language. These are "while loop" and "do-while loop". Firstly, these loops are checked to see whether they are pretest or posttest, by creating a boolean variable called pretest and initialized as true and an integer is created and assigned to a value. After that, a loop that has a boolean expression condition that does not satisfy for the created integer is created. If the loop executes, it is posttest, else it is pretest. It can be seen from the output that while loop is pretest and do-while loop is posttest in Dart. Then, break and continue statements are used to see what happens. Break statement is to terminate a loop at a point and continue statement is to terminate the iteration and the loop jumps to the next iteration. The output is given above for both statements in both loops.

LUA

CODE:

```
-- while loop test check
print("-----------CHECK PRETEST OR POSTTEST FOR WHILE LOOP-----------")
i = 3
pretest = true

-- posttest check
while i > 10 do
    pretest = false
    print("Lua is posttest for while loop.")
    i = i - 1
end

-- pretest check
if pretest == true then
    print("Lua is pretest for while loop.")
end

-- repeat until loop test check
print("-----------CHECK PRETEST OR POSTTEST FOR REPEAT UNTIL
LOOP-----------")
```

```lua
j = 3
pretest2 = true

-- posttest check
repeat
    j = j + 1
    pretest = false
    print("Lua is posttest for repeat until loop")
until j >= 3

-- pretest check
if pretest == true then
    print("Lua is pretest for repeat until loop")
end

-- break statement for while loop
print("-----------BREAK STATEMENT FOR WHILE LOOP-----------")
var = 3
loopCount = 1
print("before the loop var is: ", var)
while var < 10 do
    print("loop: ", loopCount, " var: ", var)
    var = var + 1
    loopCount = loopCount + 1
    if(var == 8) then
        break
    end
end
print("end of while loop")
print("loop used break when value was: ", var)

-- break statement for repeat until loop
print("-----------BREAK STATEMENT FOR REPEAT UNTIL LOOP-----------")
var2 = 2
print("before the loop var is: ", var2)
```

```lua
loopCount2 = 1
repeat
    print("loop: ", loopCount2, " var: ", var2)
    var2 = var2 + 1
    loopCount2 = loopCount2 + 1
    if var2 == 5 then
        break
    end
until var2 > 10
print("end of repeat until loop")
print("loop used break when value was: ", var2)

-- break statement for for loop
print("-----------BREAK STATEMENT FOR FOR LOOP-----------")
loopCount3 = 1
for i = 3, 10, 1 do
    if i == 6 then
        var3 = i
        break
    end
    print("loop: ", loopCount3, " var: ", i)
    loopCount3 = loopCount3 + 1
end
print("end of for loop")
print("loop used break when value was: ", var3)
```

OUTPUT:

```
-----------CHECK PRETEST OR POSTTEST FOR WHILE LOOP-----------
Lua is pretest for while loop.
-----------CHECK PRETEST OR POSTTEST FOR REPEAT UNTIL LOOP-----------
Lua is posttest for repeat until loop
-----------BREAK STATEMENT FOR WHILE LOOP-----------
before the loop var is:     3
loop:   1    var:   3
loop:   2    var:   4
```

```
loop:   3    var:   5
loop:   4    var:   6
loop:   5    var:   7
end of while loop
loop used break when value was:     8
----------BREAK STATEMENT FOR REPEAT UNTIL LOOP----------
before the loop var is:     2
loop:   1    var:   2
loop:   2    var:   3
loop:   3    var:   4
end of repeat until loop
loop used break when value was:     5
----------BREAK STATEMENT FOR FOR LOOP----------
loop:   1    var:   3
loop:   2    var:   4
loop:   3    var:   5
end of for loop
loop used break when value was:     6
```

Explanation:

I tested the pretest/posttest condition with two logically controlled loops in Lua: While loop and repeat until loop. I used the same way to declare whether the loops are pretest and posttest like I did for the other languages (with a boolean variable). It is seen that while loop is pretest and repeat until loop is posttest. Then, I implemented break statement for while loop, repeat until loop and for loop. Break statement is to exit from the loop at a specific point. The code displays at which point the loop uses break. There is no simple continue statement in Lua. There is a statement called goto continue which uses `::continue::` syntax. However, that statement just does not compile at any compilers I tried to use. Therefore, I did not add that notation to my code.

PHP

Code:

```
<!DOCTYPE html>
<html>
<body>

<?php
```

```php
// check pretest/posttest for while loop
echo "---------CHECK PRETEST/POSTTEST FOR WHILE LOOP---------";
echo "\n";
$var = 10;
$pretest = true;
while ($var < 0){
    $pretest = false;
    echo "PHP is posttest for While loop";
    echo "\n";
}

if($pretest == true){
    echo "PHP is pretest for While loop";
    echo "\n";
}


// check pretest/posttest for do-while loop
echo "---------CHECK PRETEST/POSTTEST FOR DO-WHILE LOOP---------";
echo "\n";
$var = 10;
$pretest = true;
do{
    $pretest = false;
    echo "PHP is posttest for While loop";
    echo "\n";
}while($var < 0);

if($pretest == true){
    echo "PHP is pretest for While loop";
    echo "\n";
}

// break statement for while loop
echo "---------BREAK STATEMENT FOR WHILE LOOP---------";
```

```php
echo "\n";
$var2 = 4;
echo "before the loop var2 is: " , $var2;
echo "\n";
$loopCount = 1;
while($var2 < 10){
    if($var2 == 8) {
        break;
    }
    echo "loop: " , $loopCount , " , var2: ", $var2;
    echo "\n";
    $var2++;
    $loopCount++;
}
echo "end of while loop";
echo "\n";
echo "loop used break when var2 was ", $var2;
echo "\n";

// break statement for do-while loop
echo "---------BREAK STATEMENT FOR DO-WHILE LOOP---------";
echo "\n";
$var3 = 3;
echo "before the loop var2 is: " , $var3;
echo "\n";
$loopCount2 = 1;
do{
    if($var3 == 7) {
        break;
    }
    echo "loop: " , $loopCount2 , " , var3: ", $var3;
    echo "\n";
    $var3++;
    $loopCount2++;
}while($var3 < 10);
```

```php
echo "end of while loop";
echo "\n";
echo "loop used break when var3 was ", $var3;
echo "\n";

// break statement for for loop
echo "---------BREAK STATEMENT FOR FOR LOOP---------";
echo "\n";
for($i = 0; $i < 10; $i++){
    echo "i: ", $i;
    echo "\n";
    if($i == 8){
        $temp = $i;
        break;
    }
}
echo "end of for loop";
echo "\n";
echo "loop used break when the value was: ", $temp;
echo "\n";

// continue statement for while loop
echo "---------CONTINUE STATEMENT FOR WHILE LOOP---------";
echo "\n";
$var4 = 3;
$loop = 1;
while($var4 < 10) {
  if ($var4 == 5) {
    echo $var4, " is skipped";
    echo "\n";
    $temp = $var4;
    $var4++;
    continue;
  }
  echo "loop: " , $loop, ", var: ",  $var4;
```

```
  echo "\n";
  $var4++;
  $loop++;
}
echo "end of while loop";
echo "\n";
echo "loop used continue when the value was: ", $temp;
echo "\n";

// continue statement for do-while loop
echo "---------CONTINUE STATEMENT FOR DO-WHILE LOOP---------";
echo "\n";
$var4 = 3;
$loop = 1;
do{
  if ($var4 == 7) {
    echo $var4, " is skipped";
    echo "\n";
    $temp = $var4;
    $var4++;
    continue;
  }
  echo "loop: " , $loop, " , var: ",  $var4;
  echo "\n";
  $var4++;
  $loop++;
}while($var4 < 10);
echo "end of do-while loop";
echo "\n";
echo "loop used continue when the value was: ", $temp;
echo "\n";


// continue statement for for loop
echo "---------CONTINUE STATEMENT FOR FOR LOOP---------";
```

```
echo "\n";
for($i = 1; $i < 9; $i++ ){
    if($i == 4){
        $temp = $i;
        echo $i, " is skipped";
        echo "\n";
        continue;
    }
    echo "i: ", $i;
    echo "\n";
}
echo "end of for loop";
echo "\n";
echo "loop used continue when the value was: ", $temp;

?>
</body>
</html>
```

Output:

```
---------CHECK PRETEST/POSTTEST FOR WHILE LOOP---------
PHP is pretest for While loop
---------CHECK PRETEST/POSTTEST FOR DO-WHILE LOOP---------
PHP is posttest for While loop
---------BREAK STATEMENT FOR WHILE LOOP---------
before the loop var2 is: 4
loop: 1 , var2: 4
loop: 2 , var2: 5
loop: 3 , var2: 6
loop: 4 , var2: 7
end of while loop
loop used break when var2 was 8
---------BREAK STATEMENT FOR DO-WHILE LOOP---------
before the loop var2 is: 3
loop: 1 , var3: 3
```

```
loop: 2 , var3: 4
loop: 3 , var3: 5
loop: 4 , var3: 6
end of while loop
loop used break when var3 was 7
---------BREAK STATEMENT FOR FOR LOOP---------
i: 0
i: 1
i: 2
i: 3
i: 4
i: 5
i: 6
i: 7
i: 8
end of for loop
loop used break when the value was: 8
---------CONTINUE STATEMENT FOR WHILE LOOP---------
loop: 1, var: 3
loop: 2, var: 4
5 is skipped
loop: 3, var: 6
loop: 4, var: 7
loop: 5, var: 8
loop: 6, var: 9
end of while loop
loop used continue when the value was: 5
---------CONTINUE STATEMENT FOR DO-WHILE LOOP---------
loop: 1 , var: 3
loop: 2 , var: 4
loop: 3 , var: 5
loop: 4 , var: 6
7 is skipped
loop: 5 , var: 8
loop: 6 , var: 9
end of do-while loop
```

```
loop used continue when the value was: 7
---------CONTINUE STATEMENT FOR FOR LOOP---------
i: 1
i: 2
i: 3
4 is skipped
i: 5
i: 6
i: 7
i: 8
end of for loop
loop used continue when the value was: 4
```

Explanation:

I tested two logically controlled loops (while and do-while) by the same method which applied to all languages in this assignment. It can be seen that while loop is pretest and do-while loop is posttest. Break statement is to exit from the loop and continue statement is to jump to the next iteration in a loop without executing the further lines in the block.

PYTHON

CODE:

```
# Online Python compiler (interpreter) to run Python online.
# Write Python 3 code in this online editor and run it.
var = 4
pretest = True
# check if python is posttest
print("-----------CHECK IF PRETEST OR POSTTEST-----------")
while(var > 10):
    var += 2
    pretest =  False
    print("Python is posttest in while loop")

# check if python is pretest
if(pretest == True):
```

```
    print("Python is pretest in while loop")


# Break statement
print("-----------BREAK STATEMENT-----------")
num = 3
loopCount = 1
while(num < 10):
    print("loop: ", loopCount, "   , num: ", num)
    num += 1
    loopCount += 1
    if(num == 6):
        break
print("end of while loop")
print("loop use break when the value is: " ,num)

# Continue statement
print("-----------CONTINUE STATEMENT-----------")
num = 2
loopCount = 1
while(num < 10):
    num += 1
    if(num == 5):
        num2 = num
        print(num , " is skipped")
        continue
    print("loop: ", loopCount, "   , num: ", num)
    loopCount += 1
print("end of while loop")
print("loop use continue when the value is: " ,num2)

# while-else statement
print("-----------WHILE-ELSE STATEMENT-----------")
num = 9
while (num > 2):
```

```
  print("num is: " ,num)
  num -= 1
else:
  print("num is less than 2")
```

Output:

```
-----------CHECK IF PRETEST OR POSTTEST-----------
Python is pretest in while loop
-----------BREAK STATEMENT-----------
loop:  1   , num:  3
loop:  2   , num:  4
loop:  3   , num:  5
end of while loop
loop use break when the value is:  6
-----------CONTINUE STATEMENT-----------
loop:  1   , num:  3
loop:  2   , num:  4
5  is skipped
loop:  3   , num:  6
loop:  4   , num:  7
loop:  5   , num:  8
loop:  6   , num:  9
loop:  7   , num:  10
end of while loop
loop use continue when the value is:  5
-----------WHILE-ELSE STATEMENT-----------
num is:  9
num is:  8
num is:  7
num is:  6
num is:  5
num is:  4
num is:  3
```

```
num is less than 2
```

Explanation:

I used while loops to make operations. According to the test code that I wrote, while loops in Python are pretest, I checked it using a boolean variable. Furthermore, I added three statements: Break, Continue and While-else. Break statement is to exit from the loop at a specific point, continue statement is to exit from an iteration and jump to the next iteration and while-else statement is to run a block of code after the loop ends.

RUBY

Code:

```ruby
# checking for pretest/posttest in loop and break statement in loop
puts "---------CHECK PRETEST/POSTTEST IN LOOP---------"
pretest = true

# check if posttest
loop do
  pretest = false
  puts "Ruby is posttest in loop"
  puts "now loop will use break"
  break
end
puts "loop has ended"

# check if pretest
if pretest ==  true then
    puts "Ruby is pretest in loop"
end


# checking for pretest/posttest in while loop
puts "---------CHECK PRETEST/POSTTEST IN WHILE LOOP---------"
```

```ruby
pretest2 = true
var = 12
loopCount = 1
while var < 10 do
    pretest2 = false
    puts "Ruby is posttest in while loop"
end

if pretest2 == true then
    puts "Ruby is pretest in while loop"
end

# checking for pretest/posttest in until loop
puts "---------CHECK PRETEST/POSTTEST IN UNTIL LOOP---------"
val = 3
pretest_until = true
until val > 2  do
   puts "val = #{val}"
   val +=1;
   pretest_until = false
   puts "Ruby is posttest in until loop"
end

if pretest_until == true then
    puts "Ruby is pretest in until loop"
end

# Break statement in while loop
puts "---------BREAK STATEMENT IN WHILE LOOP---------"
var2 = 1
puts "before the loop var is: #{var2}"
loopCount = 1
while var2 < 10 do
    if var2 == 7 then
        break
```

```
    end
    puts "loop: #{loopCount} var: #{var2} "
    var2 = var2 + 1
    loopCount = loopCount + 1
end
puts "end of while loop"
puts "loop uses break when the var was: #{var2}"

# Break statement in until loop
puts "---------BREAK STATEMENT IN UNTIL LOOP---------"
var3 = 5
puts "before the loop var is: #{var3}"
loopCount2 = 1
until var3 > 12 do
    if var3 == 9 then
        break
    end
    puts "loop: #{loopCount2} var: #{var3}"
    var3 = var3 + 1
    loopCount2 = loopCount2 + 1
end
puts "end of until loop"
puts "loop used break when the var was: #{var3}"

# Break statement in for loop
puts "---------BREAK STATEMENT IN FOR LOOP---------"
for i in 0..10
   if i == 6 then
       var4 = i
       break
   end
   puts "var: #{i}"
end
puts "end of for loop"
puts "loop used break when the var was: #{var4}"
```

```ruby
# next statement in while loop
puts "---------NEXT STATEMENT IN WHILE LOOP---------"
var5 = 1
puts "before the loop var is: #{var5}"
loopCount3 = 1
while var5 < 11 do
    var5 = var5 + 1

    if var5 == 7 then
        temp = var5
        puts "value #{var5} is skipped"
        next
    end
    puts "loop: #{loopCount3} var: #{var5} "
    loopCount3 = loopCount3 + 1
end
puts "end of while loop"
puts "loop uses continue when the var was: #{temp}"


# next statement in until loop
puts "---------NEXT STATEMENT IN UNTIL LOOP---------"
var6 = 3
puts "before the loop var is: #{var6}"
loopCount4 = 1
until var6 > 9 do
    var6 = var6 + 1
    if var6 == 4 then
        temp = var6
        puts "value #{var6} is skipped"
        next
    end
    puts "loop: #{loopCount4} var: #{var6} "
    loopCount4 = loopCount4 + 1
```

```ruby
end
puts "end of until loop"
puts "loop uses continue when the var was: #{temp}"


#next statement in for loop
puts "---------NEXT STATEMENT IN FOR LOOP---------"
for i in 0..9
    if i == 5 then
        temp = i
        puts "value #{i} is skipped"
        next
    end
    puts "value: #{i}"
end
puts "end of for loop"
puts "loop uses continue when the var was: #{temp}"
```

Output:

```
ruby main.rb
---------CHECK PRETEST/POSTTEST IN LOOP---------
Ruby is posttest in loop
now loop will use break
loop has ended
---------CHECK PRETEST/POSTTEST IN WHILE LOOP---------
Ruby is pretest in while loop
---------CHECK PRETEST/POSTTEST IN UNTIL LOOP---------
Ruby is pretest in until loop
---------BREAK STATEMENT IN WHILE LOOP---------
before the loop var is: 1
loop: 1 var: 1
loop: 2 var: 2
loop: 3 var: 3
loop: 4 var: 4
```

```
loop: 5 var: 5
loop: 6 var: 6
end of while loop
loop uses break when the var was: 7
---------BREAK STATEMENT IN UNTIL LOOP---------
before the loop var is: 5
loop: 1 var: 5
loop: 2 var: 6
loop: 3 var: 7
loop: 4 var: 8
end of until loop
loop used break when the var was: 9
---------BREAK STATEMENT IN FOR LOOP---------
var: 0
var: 1
var: 2
var: 3
var: 4
var: 5
end of for loop
loop used break when the var was: 6
---------NEXT STATEMENT IN WHILE LOOP---------
before the loop var is: 1
loop: 1 var: 2
loop: 2 var: 3
loop: 3 var: 4
loop: 4 var: 5
loop: 5 var: 6
value 7 is skipped
loop: 6 var: 8
loop: 7 var: 9
loop: 8 var: 10
loop: 9 var: 11
end of while loop
loop uses continue when the var was: 7
```

```
---------NEXT STATEMENT IN UNTIL LOOP---------
before the loop var is: 3
value 4 is skipped
loop: 1 var: 5
loop: 2 var: 6
loop: 3 var: 7
loop: 4 var: 8
loop: 5 var: 9
loop: 6 var: 10
end of until loop
loop uses continue when the var was: 4
---------NEXT STATEMENT IN FOR LOOP---------
value: 0
value: 1
value: 2
value: 3
value: 4
value 5 is skipped
value: 6
value: 7
value: 8
value: 9
end of for loop
loop uses continue when the var was: 5
```

Explanation:

In Ruby language I tested three logically controlled loops(loop, until loop, while loop) to see whether they are pretest or posttest by using the same method with all languages in this homework. Only the simple is posttest, and while loop and until loop are pretest. Break statement is to exit from a loop at a specific point and next statement is the equivalent of the continue statement in Dart, Python etc. in Ruby. Corresponding outputs and comment are given above.

RUST

Code:

```
fn main() {
    // check if while loop pretest/posttest
    println!("----------CHECK PRETEST/POSTTEST FOR WHILE
LOOP----------");
    let mut n = 0;
    print!("The value of n before loop: ");
    println!("{}", n);
    let mut pretest = true;
    while n > 10{
        pretest = false;
        println!("Rust is posttest for while loop");
    }

    if pretest{
        println!("Rust is pretest for while loop");
    }

    // break statement for while loop
    println!("----------BREAK STATEMENT FOR WHILE LOOP----------");
    let mut x = 1;
    let mut loop_count = 1;
    print!("The value of x before loop: ");
    println!("{}", x);
    while x < 16{
        if x == 10{
            break;
        }
        print!("loop: ");
        print!("{}", loop_count);
        print!(" , x: ");
```

```
        println!("{}", x);
        x = x + 1;
        loop_count = loop_count + 1;
    }
    println!("end of while loop");
    print!("loop used break when x was: ");
    println!("{}", x);

    // continue statement for while loop
    println!("----------CONTINUE STATEMENT FOR WHILE LOOP----------");
    let mut y = 1;
    let mut loop_count2 = 1;
    let mut temp = 0;
    print!("The value of x before loop: ");
    println!("{}", y);
    while y < 10{
        y = y + 1;
        loop_count2 = loop_count2 + 1;
        if y == 6{
            temp = y;
            print!("{}", y);
            println!(" is skipped");
            continue;
        }
        print!("loop: ");
        print!("{}", loop_count2);
        print!(" , y: ");
        println!("{}", y);

    }
    println!("end of while loop");
    print!("loop used continue when y was: ");
    println!("{}", temp);
}
```

```
Output:
----------CHECK PRETEST/POSTTEST FOR WHILE LOOP----------
The value of n before loop: 0
Rust is pretest for while loop
----------BREAK STATEMENT FOR WHILE LOOP----------
The value of x before loop: 1
loop: 1 , x: 1
loop: 2 , x: 2
loop: 3 , x: 3
loop: 4 , x: 4
loop: 5 , x: 5
loop: 6 , x: 6
loop: 7 , x: 7
loop: 8 , x: 8
loop: 9 , x: 9
end of while loop
loop used break when x was: 10
----------CONTUNIE STATEMENT FOR WHILE LOOP----------
The value of x before loop: 1
loop: 2 , y: 2
loop: 3 , y: 3
loop: 4 , y: 4
loop: 5 , y: 5
6 is skipped
loop: 7 , y: 7
loop: 8 , y: 8
loop: 9 , y: 9
loop: 10 , y: 10
end of while loop
loop used continue when y was: 6
```

Explanation:

In Rust, I used only while loop. While loop is pretest. Break statement is to exit from the loop and continue statement is to jump to the next iteration without executing further lines in the block.

2) Evaluating these languages, I can say that most of them are very similar to each other in terms of syntax and statement purposes. "break" syntax always means the same for all languages. In my opinion, most readable and most writable language for loops is Python since there are less keywords. Most of the languages have "do, end, then" keywords that

harms the writability. For readability, what makes Python good is that there are less symbols while defining or printing variables such as ($, #{}, {} etc.) The only problem with Python is that Python does not have do-while loop which iterates through the loop once no matter what. I think that is a design issue and developers, perhaps, did it intentionally but even if I did some research about it I did not find any satisfying answers. One think that is a big issue for me is that Lua does not have a simple continue statement and I think that is very problematic in terms of writability. Also, in Ruby, there is not a continue statement but there is a next statement, which is actually a better name, considering the purpose of this statement. In terms of syntax, all languages except Python only differ each other by very small differences.

3) I used online compilers, online resources and lecture slides. I, similarly to homework 1, did this homework not language by language but I did it operation by operation. Since most of the structures are very similar to each other among all languages, this was not as fun as homework 1. Differently from the first homework, this time what I have done is to look for the syntaxes and special cases and write my code according to them. The URLs for the online compilers/interpreters that I used are as follows,

Dart: dartpad.dev
Javascript:
Lua: https://repl.it/languages/lua
PHP: https://paiza.io/en/projects/new
Phyton: https://www.programiz.com/python-programming/online-compiler/
Ruby: https://repl.it/languages/ruby
Rust: https://play.rust-lang.org