Report

In our code, we utilized a function call after an `AND` operator to determine whether or not short-circuit evaluation is used in a given programming language. If the language uses short-circuit evaluation, the `print_answer()` function will not be executed and the output will be "`If you only see this then we DO have short circuit evaluation`". However, if the language does not use short-circuit evaluation, the `print_answer()` function will be executed and the output will be "`We DO NOT have short-circuit evaluation!`" followed by "`If you only see this then we DO have short circuit evaluation`". By placing `print_answer()` as the second operand in the `AND` expression, we took advantage of the fact that the second operand will only be evaluated if the first operand is true. In this way, we were able to determine whether or not short-circuit evaluation is used in a given programming language.

Language	Short-Circuit?	Output
Ada	No	We DO NOT have short-circuit evaluation! If you only see this then we DO have short circuit evaluation
Bash	Yes	If you only see this then we DO have short circuit evaluation
Perl	Yes	If you only see this then we DO have short circuit evaluation
PHP	Yes	If you only see this then we DO have short circuit evaluation

In the Ada example, we're using the `AND` operator and `print_answer` function, but it does not demonstrate short-circuit evaluation since the second operand will always be evaluated in Ada. I think it's worth noting that Ada does have an additional operator called `AND then` which does have short circuit evaluation.

In the Bash, Perl, and PHP examples, we are using the `&&` operator, which demonstrates short-circuit evaluation. The `print_answer` function is only called if the first operand is true, which allows the code to avoid unnecessary evaluations of the second operand if it is not needed to determine the result of the expression.