프로그래밍언어 HW #4 20191629 이주헌

1. Show the order of evaluation.

① $a * b - 1 + c \rightarrow (((a * b)' - 1)^{2} + c)^{3}$

② $a * (b - 1) / c \text{ mod } d \rightarrow ((a * (b - 1)' / c)^{2} \text{ mod } d)^{3}$

③ $(a - b) / c \ \& \ (d * e / a - 3) \rightarrow ((((a - b)^{1} / c)^{2} \& ((d * e)^{3} / a)^{4})^{5} - 3)^{6}$

④ $-a \text{ or } c = d \text{ and } e \rightarrow ((-a)' \text{ or} ((c = d)^{2} \text{ and } e)^{3})^{4}$

⑤ $a > b \text{ xor } c \text{ or } d <= 17 \rightarrow (((a > b)' \text{ xor } c)^{2} \text{ or } (d <= 17)^{3})^{4}$

⑥ $-a + b \rightarrow (-(a + b)')^{2}$

2. Show the order of evaluation, assuming no precedence rules present and right-associativity.

① $a * b - 1 + c \rightarrow (a * (b - (1 + c)')^{2})^{3}$

② $a * (b - 1) / c \text{ mod } d \rightarrow (a * ((b - 1)^{2} / (c \text{ mod } d)')^{3})^{4}$

③ $(a - b) / c \ \& \ (d * e / a - 3) \rightarrow ((a - b)^{5} / (c \ \& \ (d * (e / ((a - 3)')^{2})^{3})^{4})^{6}$

④ $-a \text{ or } c = d \text{ and } e \rightarrow (-(a \text{ or } (c = (d \text{ and } e)')^{2})^{3})^{4}$

⑤ $a > b \text{ xor } c \text{ or } d <= 17 \rightarrow (a > (b \text{ xor } (c \text{ or } (d <= 17)')^{2})^{3})^{4}$

⑥ $-a + b \rightarrow (-(a + b)')^{2}$

3. Let the function "fun" defined. What are the values of sum1 and sum2...

① left-associative?

i) sum1 → i/2 eval, fun(&i) eval. → 5 + 41 = 46.

↳ 46.

ii) sum2 → fun(&j) eval, j/2 eval. → 41 + 7 = 48

↳ 48

② right-associative?

i) sum1 → fun(&i) eval, i/2 eval → 41 + 7 = 48

↳ 48

ii) sum 2 → j/2 eval, fun(&j) eval → 5 + 41 = 46

↳ 46

4. Consider a C program. What is the value of x if...

① left-associative?

assign x to x, eval fun(&x) → x=3, x=8

x=8

② right-associative?

eval fun(&x), add x, assign to x → x becomes 8, 4+8 = 12, x=12

x=12.

5. Let a function "fun". Explain the result.

There are 2 prints of "b". Let the first print be case (i), and the second (ii). Here, we will assume that the assignment operator is right-associative and the addition operator is left-associative.

In case (i), a is evaluated and fun() is evaluated, then assigned to b. Since a is evaluated before the call to fun(), b is 10 + fun(). The function fun() transforms the value of a to 20 and returns it. Thus, b = 10 + 20; essentially, 30.

In case (ii), fun() is evaluated before a is evaluated. In the call to fun(), a is transformed to 20, and the subsequent evaluation of a returns 20. Unlike in case (i), b = 20 + 20, which means b = 40.

The function fun() is a function with side effect that modifies the global state a. Therefore, the order of evaluation becomes significant.