Tony Maldonado
CS 471
October 20, 2020

Chapter 7 Problem Set

Question 9

9. Assume the following rules of associativity and precedence for expressions:

Precedence	*Highest*	*, /, **not***
		+ , − , &, **mod**
		- (unary)
		=, /=, < , <=, >=, >
		and
	Lowest	**or**, **xor**
Associativity	*Left to right*	

Show the order of evaluation of the following expressions by parenthesizing all subexpressions and placing a superscript on the right parenthesis to indicate order. For example, for the expression

```
a + b * c + d
```

the order of evaluation would be represented as

```
((a + (b * c)1)2 + d)3
```

a. `a * b - 1 + c`
b. `a * (b - 1) / c` **mod** `d`
c. `(a - b) / c & (d * e / a - 3)`
d. `-a` **or** `c = d` **and** `e`
e. `a > b` **xor** `c` **or** `d <= 17`
f. `-a + b`

Answers:

a. a * b – 1 + c
(((a * b)^1 – 1)^2 + c)^3

b. a * (b – 1) / c mod d
(((a * (b – 1)^1)^2 / c)^3 mod d)^4

c. (a – b) / c & (d * e / a – 3)
(((a – b)^1 / c)^5 & (((d * e)^2 / a)^3 – 3)^4)^6

d. -a or c = d and e
((-a)^1 or ((c = d)^2 and e)^3)^4

e. a > b xor c or d <= 17
(((a > b)^1 xor c)^3 or (d <= 17)^2)^4

f. -a + b
(-(a + b)^1)^2

Tony Maldonado
CS 471
October 20, 2020

Question 13

13. Let the function `fun` be defined as

```
int fun(int*k) {
 *k += 4;
 return 3 * (*k) - 1;
 }
```

Suppose `fun` is used in a program as follows:

```
void main() {
  int i = 10, j = 10, sum1, sum2;
  sum1 = (i / 2) + fun(&i);
  sum2 = fun(&j) + (j / 2);
}
```

What are the values of `sum1` and `sum2`

a. operands in the expressions are evaluated left to right?
b. operands in the expressions are evaluated right to left?

Answers:

a. sum1 = 46
sum2 = 48

b. sum1 = 48
sum2 = 46

Tony Maldonado
CS 471
October 20, 2020

Question 19

19. Consider the following C program:

```
int fun(int *i) {
  *i += 5;
  return 4;
}
void main() {
  int x = 3;
  x = x + fun(&x);
  }
```

What is the value of x after the assignment statement in main, assuming

a. operands are evaluated left to right.
b. operands are evaluated right to left.

Answers:

a. x = 7
b. x = 12