Jake Herson

October 13, 2016

CS4121 Homework Assignment #3

1. Without the parenthesis, the two operations could equal many different numbers when added together. (* 2 3 4 5 (- 16 9 4)) = 360 whereas (*2 3 4 5 (- 16 9) 4) = 3360. When parenthesis are added they describe the order of operations and associative property perfectly without fail.

2. No. If i is a double pointer, ie: int **i, then &i should give the address of the pointer *i, while &(&i) should theoretically give the address of that address. However, an address itself does not have an address because it is a compile time construct. The compiler would most likely complain.

3. The following code creates a short-circuit evaluation.

```
(1) A and then B
        if A then
                B
        else
                false
(2) A or else B
        if A then
                true
        else
                B
```

4. If a is zero, the answer depends on b: if b is zero, we get a floating-point exception; if b is non-zero, the answer is zero. If b is zero, we get a floating-point exception, regardless of the value of a.

   If we were to design a language in which the expression is guaranteed to evaluate to false when either a or b is false, that would mean relaxing the requirement that the operands of the && operator be evaluated from left to right. Then, both operands would have to be evaluated, because we don't know a priori which one (if any) will evaluate to false. This would defeat the purpose of the short-circuit operator &&, which is to avoid evaluating the second operand, if the first one yields false.

   In addition, if the compiler decides which of the operands is to be evaluated first, there is always the risk that the operand chosen to be evaluated first will fail. This would defeat the purpose for what we seek, which is to guarantee that the expression yield false if either a or b (but not both) is zero.

5.

```
do {
        line = read_line();
        if (!all_blanks(line)) consume_line(line);
} while !(all_blanks(line));
```

6. Nope, not convincing. I believe this would be a good alternative in C.

```
int first_zero_row = -1;
int i, j;
for (i = 0; i < n; i++) {
        int all_zeroes = 1;
        for (j = 0; j < n; j++) {
                if (A[i][j]) {
                        all_zeroes = 0;
                        break;
                }
        }
        if (all_zeroes) {
                first_zero_row = i;
                break;
        }
}
```

7.