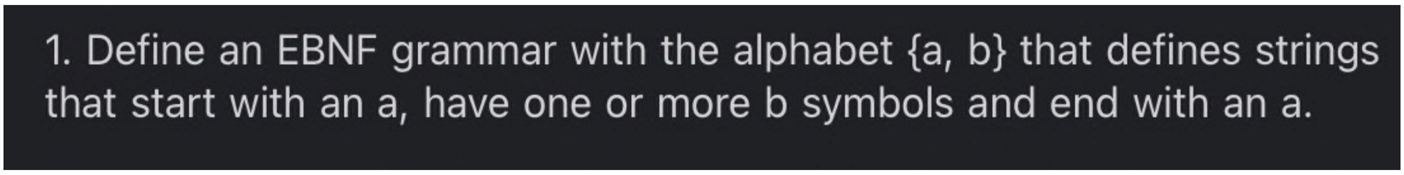
Matthew Barg
9/15/21


1. Define an EBNF grammar with the alphabet {a, b} that defines strings that start with an a, have one or more b symbols and end with an a.

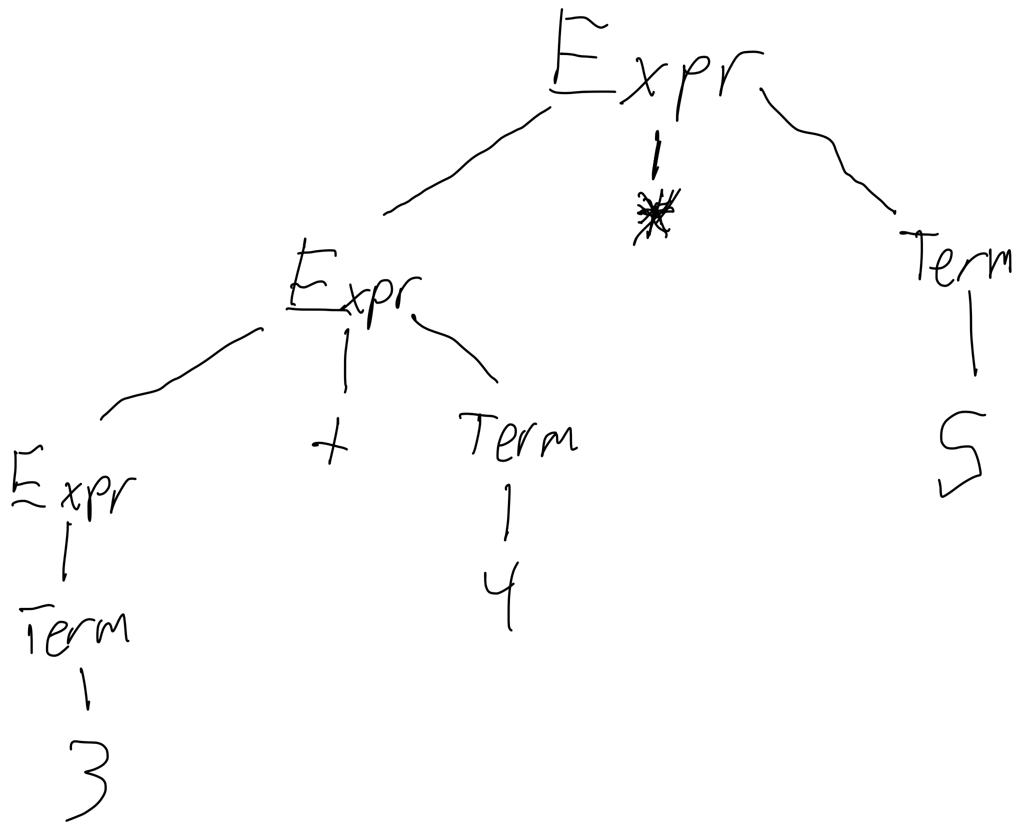
string $\rightarrow$ a (b) a

or if ( ) not supported

string $\rightarrow$ ab{b} a

2. Using each of the following grammars, draw a parse tree for 3 + 4 * 5.

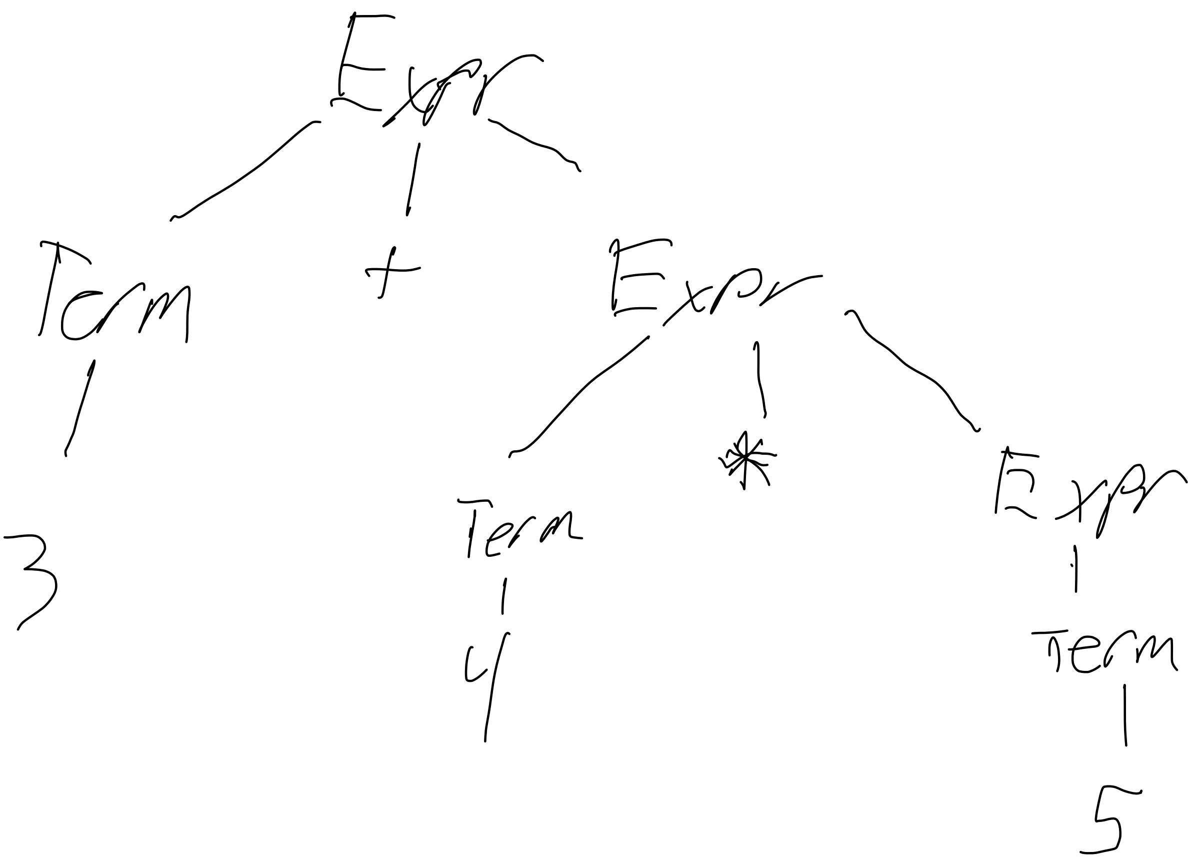
a. Expr -> Expr + Term | Expr * Term | Term
Term -> 0 | ... | 9

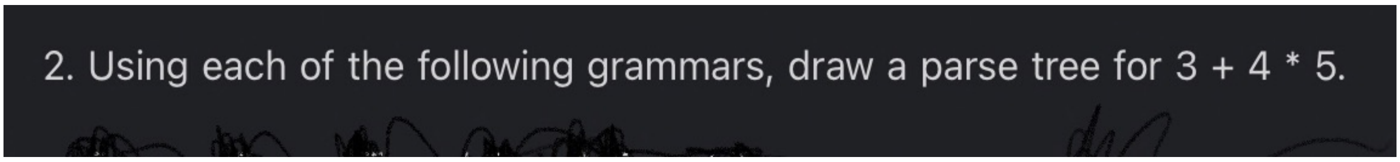


2. Using each of the following grammars, draw a parse tree for 3 + 4 * 5.

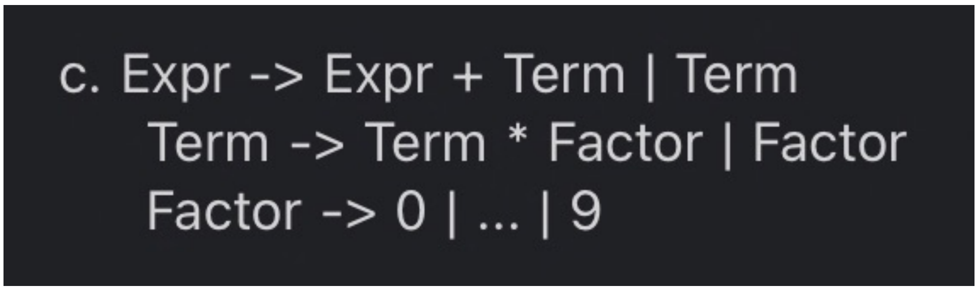
[redacted]

b. Expr -> Term + Expr | Term * Expr | Term
Term -> 0 | ... | 9

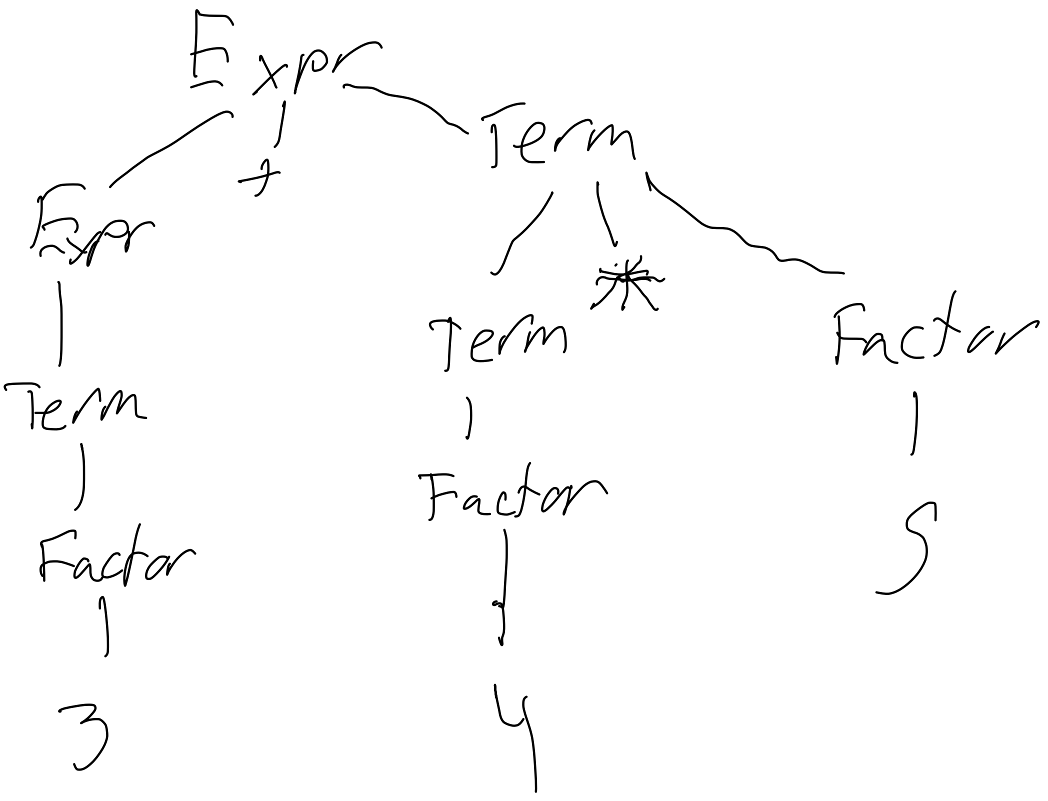


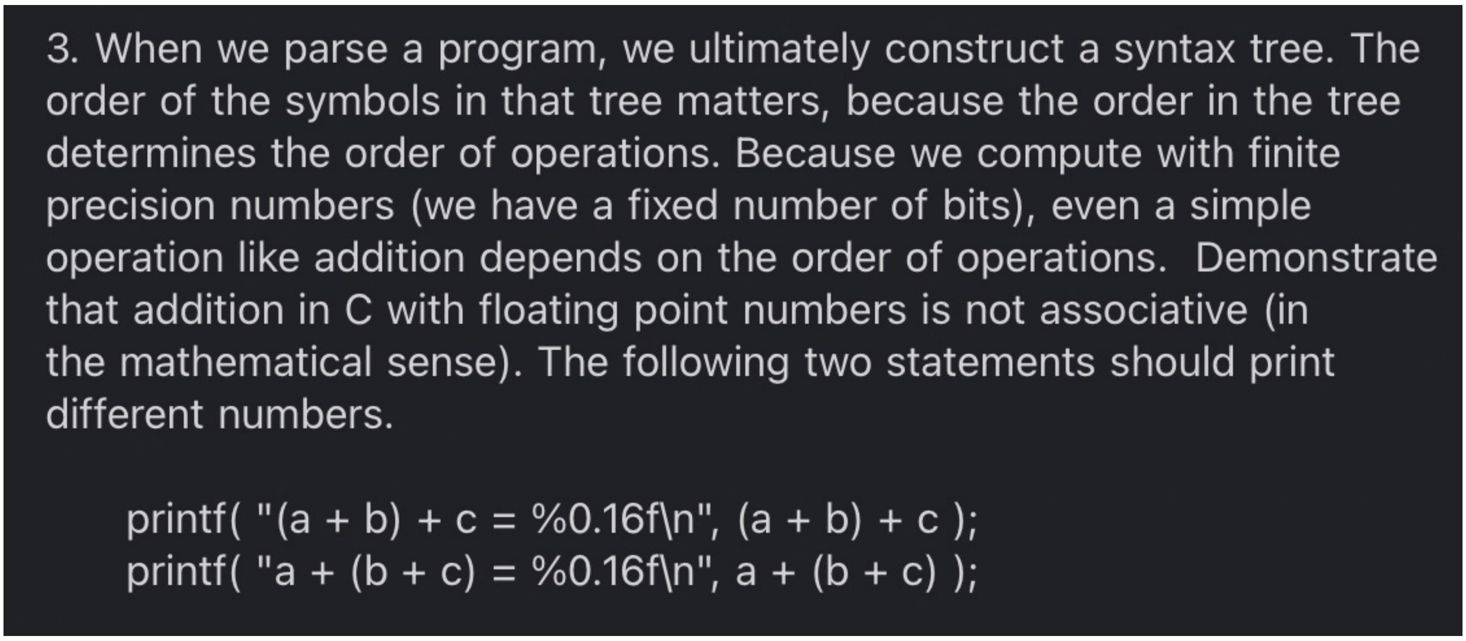


2. Using each of the following grammars, draw a parse tree for 3 + 4 * 5.



c. Expr -> Expr + Term | Term
Term -> Term * Factor | Factor
Factor -> 0 | ... | 9





3. When we parse a program, we ultimately construct a syntax tree. The order of the symbols in that tree matters, because the order in the tree determines the order of operations. Because we compute with finite precision numbers (we have a fixed number of bits), even a simple operation like addition depends on the order of operations. Demonstrate that addition in C with floating point numbers is not associative (in the mathematical sense). The following two statements should print different numbers.

```
printf( "(a + b) + c = %0.16f\n", (a + b) + c );
printf( "a + (b + c) = %0.16f\n", a + (b + c) );
```

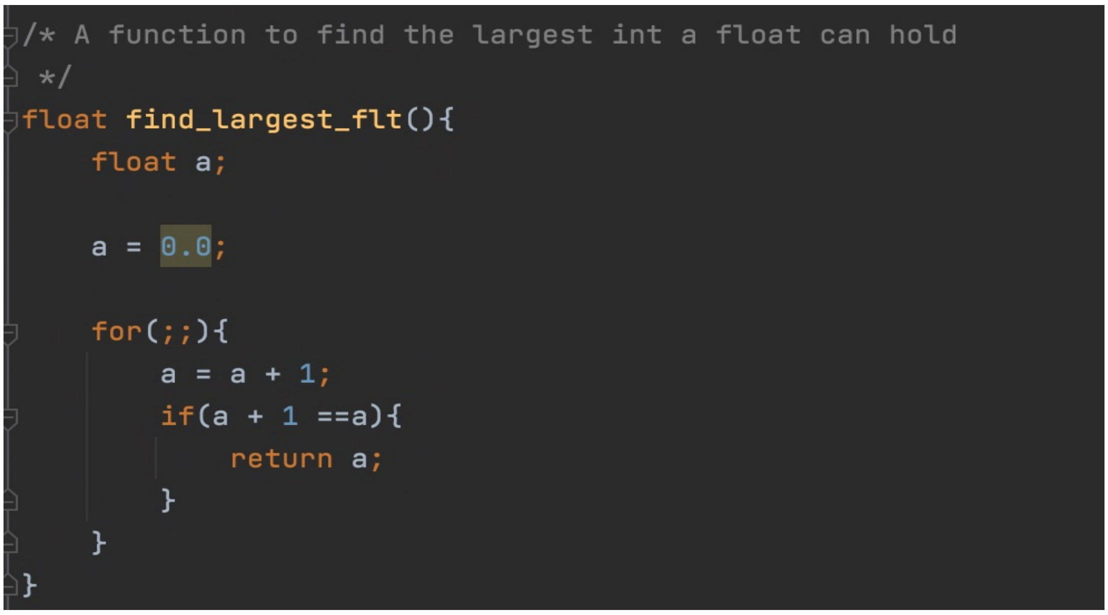
a should be 16777216

b should be -16777216

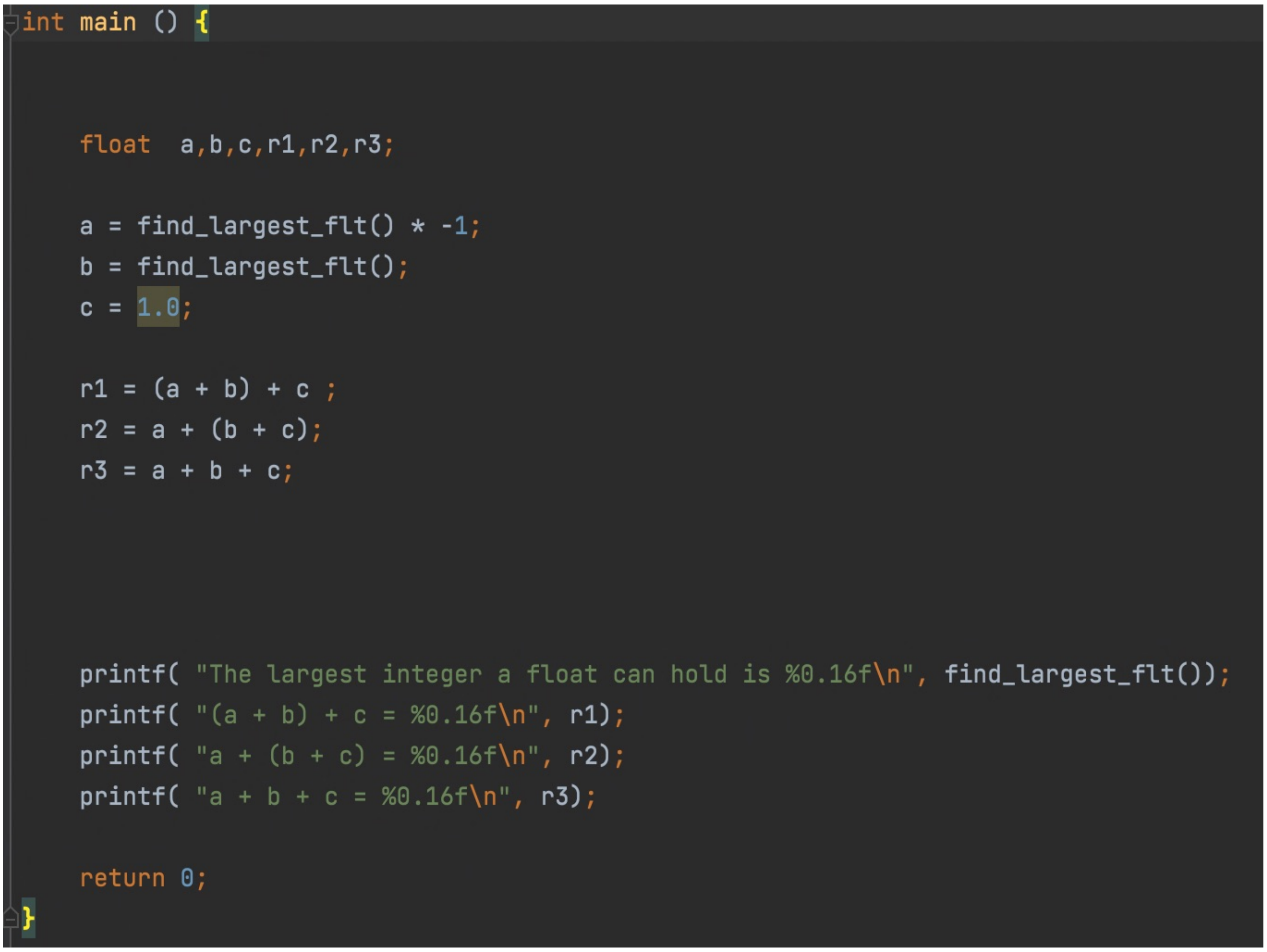
c should be 1

Also since a+b+c = (a+b)+c I can tell my c code is parsed from left to right.

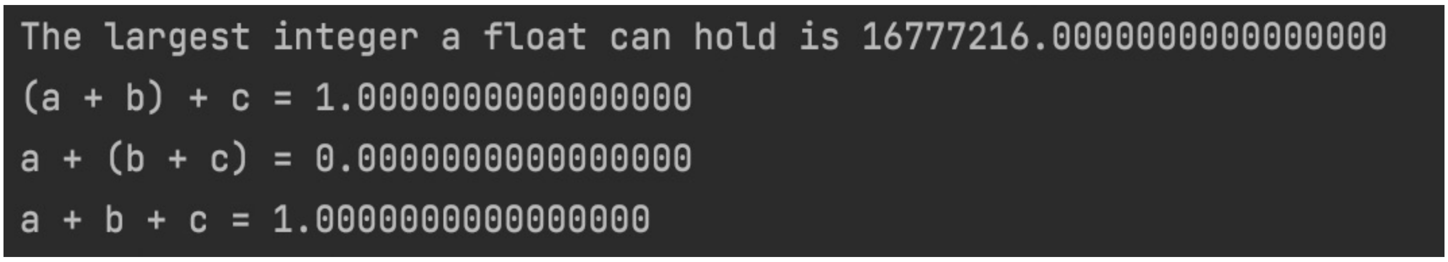
I got all these conclusions from the code below (and attached)



```
/* A function to find the largest int a float can hold
 */
float find_largest_flt(){
    float a;

    a = 0.0;

    for(;;){
        a = a + 1;
        if(a + 1 ==a){
            return a;
        }
    }
}
```



```c
int main () {


    float  a,b,c,r1,r2,r3;

    a = find_largest_flt() * -1;
    b = find_largest_flt();
    c = 1.0;

    r1 = (a + b) + c ;
    r2 = a + (b + c);
    r3 = a + b + c;




    printf( "The largest integer a float can hold is %0.16f\n", find_largest_flt());
    printf( "(a + b) + c = %0.16f\n", r1);
    printf( "a + (b + c) = %0.16f\n", r2);
    printf( "a + b + c = %0.16f\n", r3);

    return 0;
}
```



```
The largest integer a float can hold is 16777216.0000000000000000
(a + b) + c = 1.0000000000000000
a + (b + c) = 0.0000000000000000
a + b + c = 1.0000000000000000
```