Omar Mustardo & Zach Fletcher
Programming - C
Week 3/4 Project
Scan, Parse, and Evaluate

Abstract:

This program takes in and solves arithmetic expressions. It uses three main parts: the scanner, parser, and evaluator.

Scanner:

The scanner is given a file to get data from. It retrieves one character at a time and puts the characters into nodes in a linked list based on the following deterministic finite automaton.

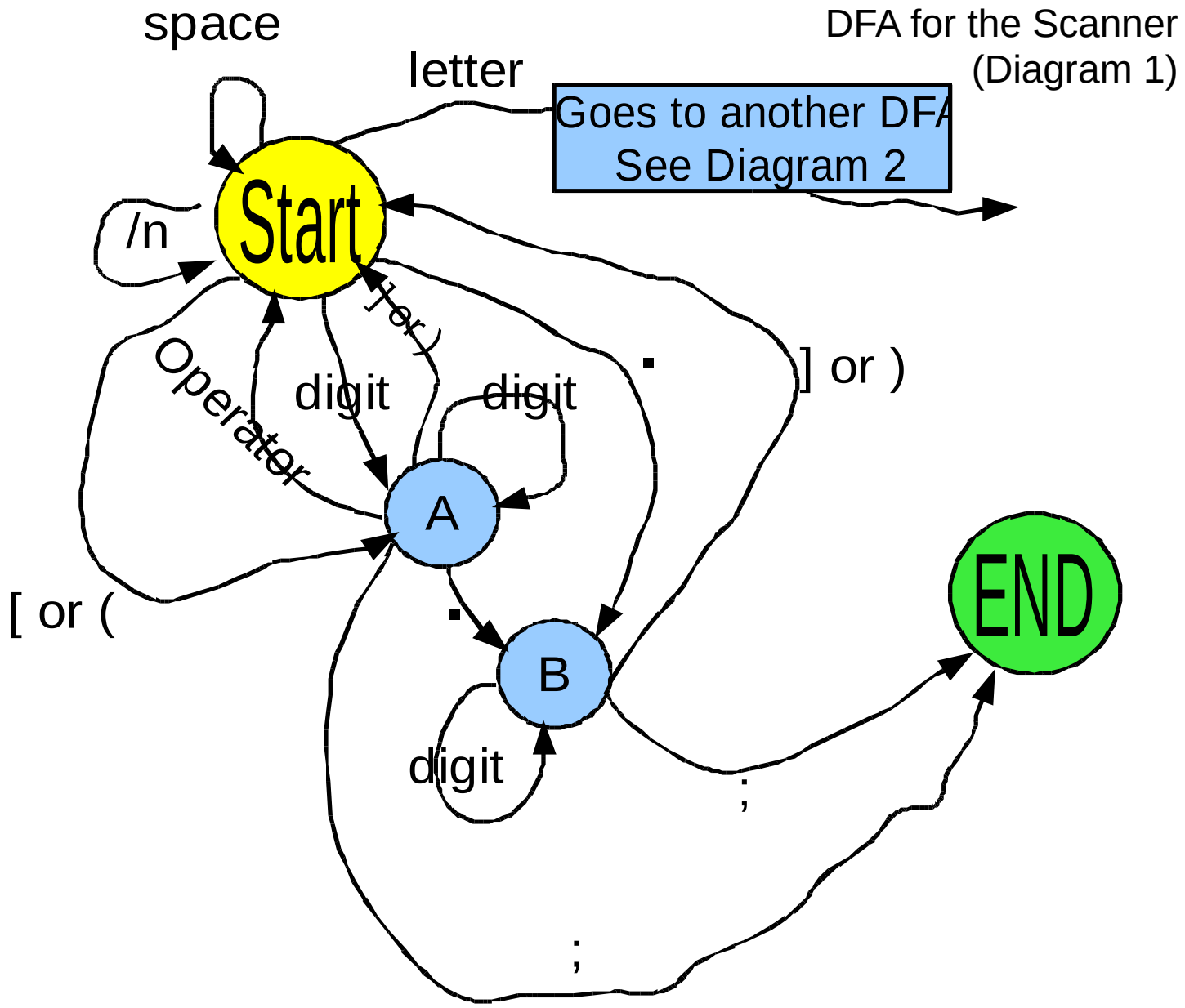
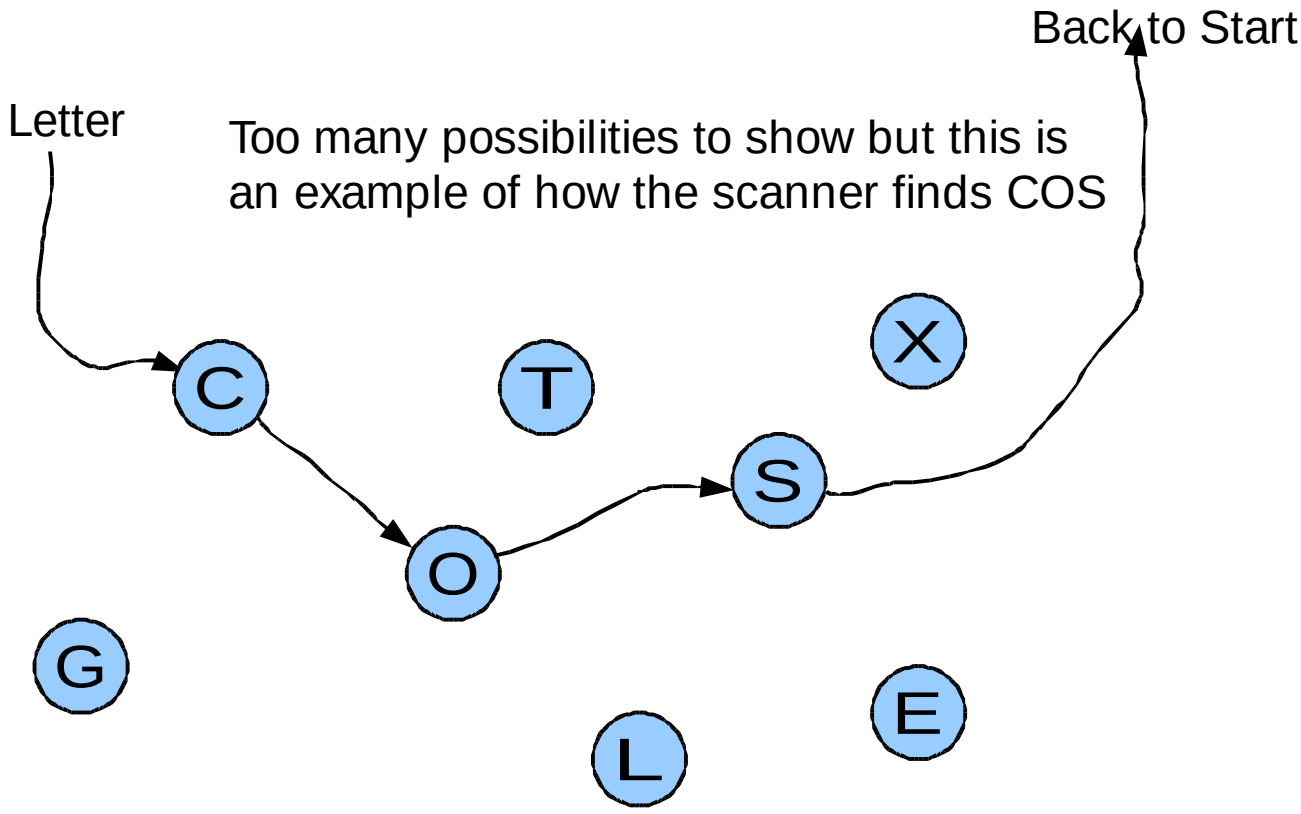


Diagram 2



Parser:

The parser takes in the head of the linked list from the Scanner. It creates a parse tree by taking the head node off of the linked list and putting it onto the tree. This parse tree can be viewed by uncommenting line 23 in main.c . We decided not to show the parse tree normally because a normal user generally doesn't want to see how their equations are computed.

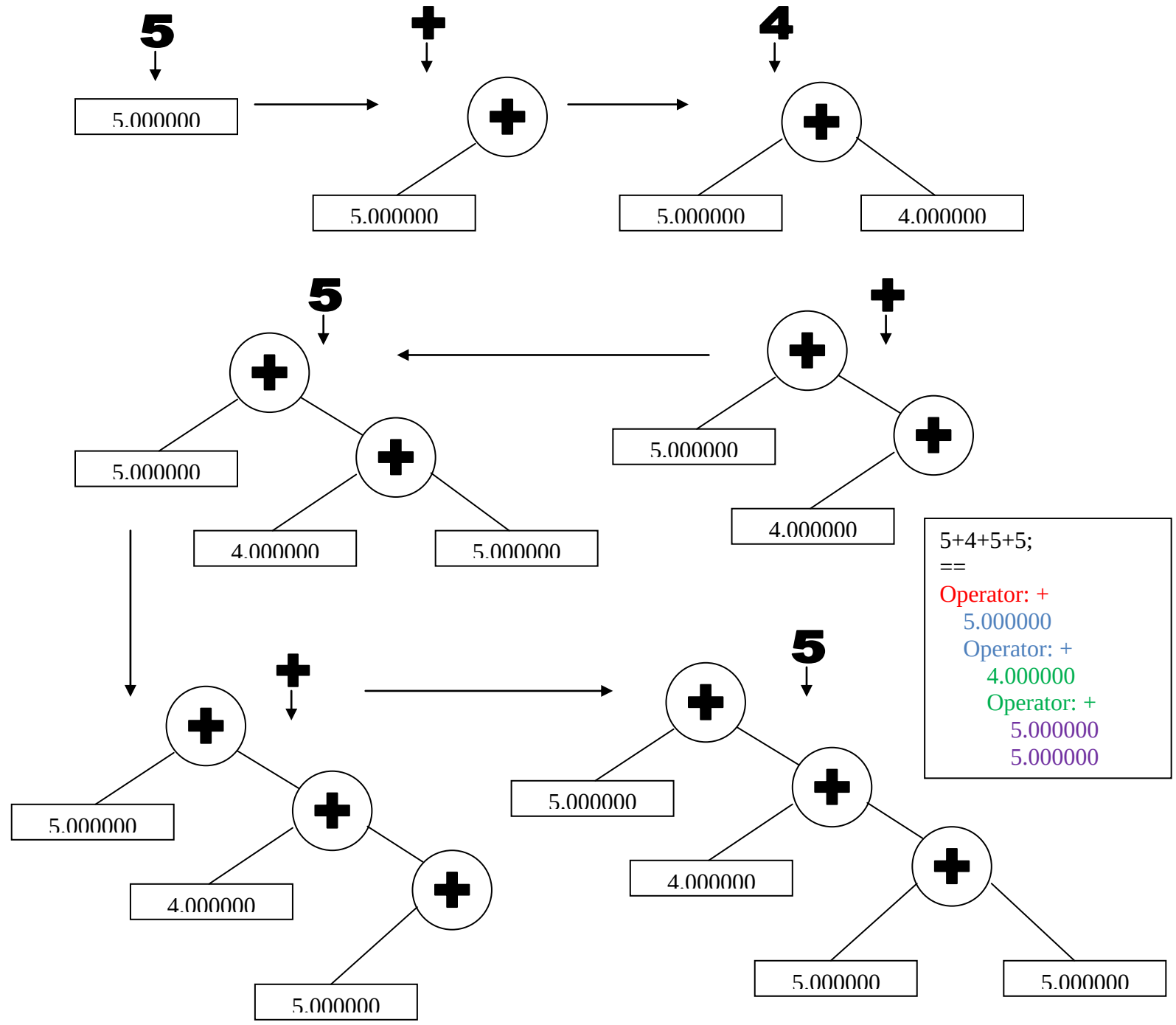
The position on the tree is determined by the typeID of the node for operators. The typeID is an integer that represents an operation. It is also used to represent numbers, parenthesis, and special functions.

typeID : representation

0 - ^ 1 - / 2 - * 3 - - 4 - + 10 - Numbers 20 - [(21 -])	30 - SIN 31 - COS 32 - TAN 33 - SQRT 34 - LN 35 - LOG 36 - EXP

A higher typeID for an operator signifies lower precedence so a node with low precedence is moved up the tree.
Parentheses are put into the tree by recursively calling the method to make a tree and assigning the head node of the recursive tree to the place where it would go in the main tree if it were a number.

Diagram 3:
Building the parse tree



The special operations are evaluated in almost the same manner as parentheses with a recursive make tree call. The difference is that special operations are evaluated and their operation is performed immediately before they are added to the main tree as a number.

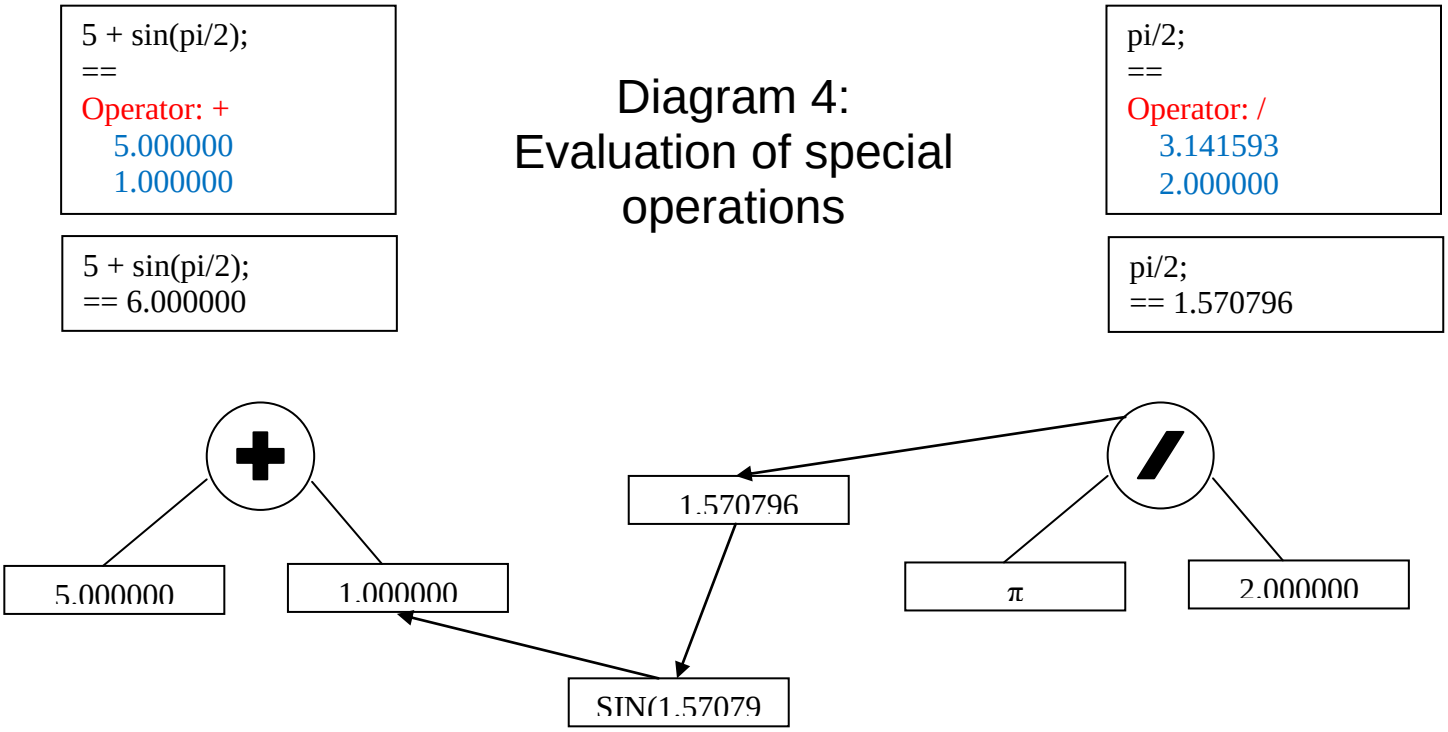


Diagram 4:
Evaluation of special operations

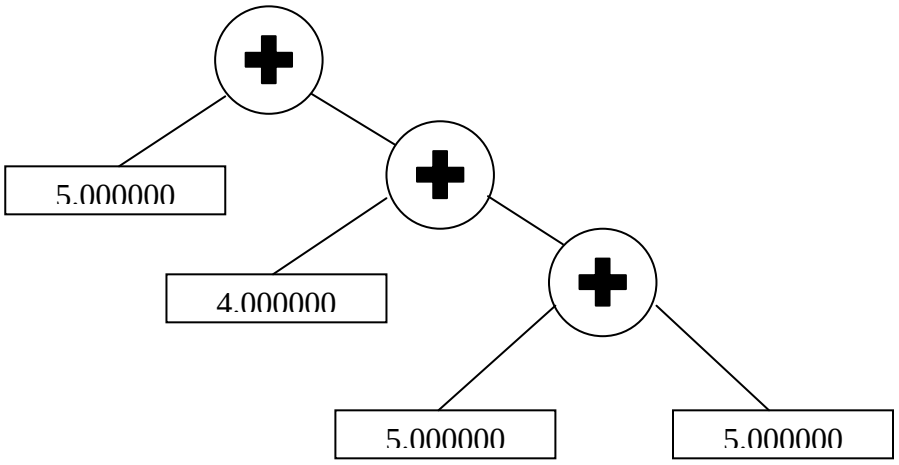
Evaluator:

The evaluation process is simple with the parse tree that we designed. The evaluator recurses to the left until it finds two numbers to perform an operation on. It performs the operation and sets the typeID of the operation's node to 10 which means that it is now the node of a number. Then it changes the value of the node to the result of the operation. Then it continues recursing through the tree until the head becomes a number at which point the evaluator returns that number.

Diagram 5 shows how a parse tree is evaluated.

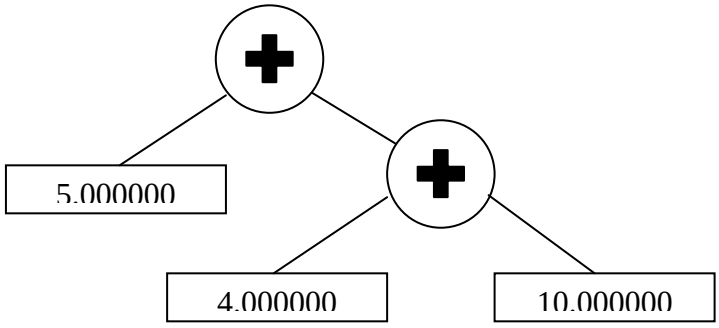
See diagram 6 for example output and visual representations of parse trees.

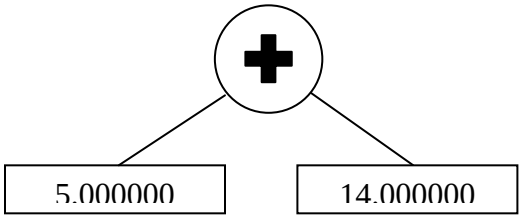
Diagram 5:
Evaluation of the parse tree




```
5+4+5+5;
==
Operator: +
   5.000000
   Operator: +
      4.000000
      Operator: +
         5.000000
         5.000000
```

5+4+5+5;
== 19.000000





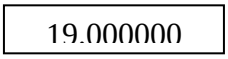
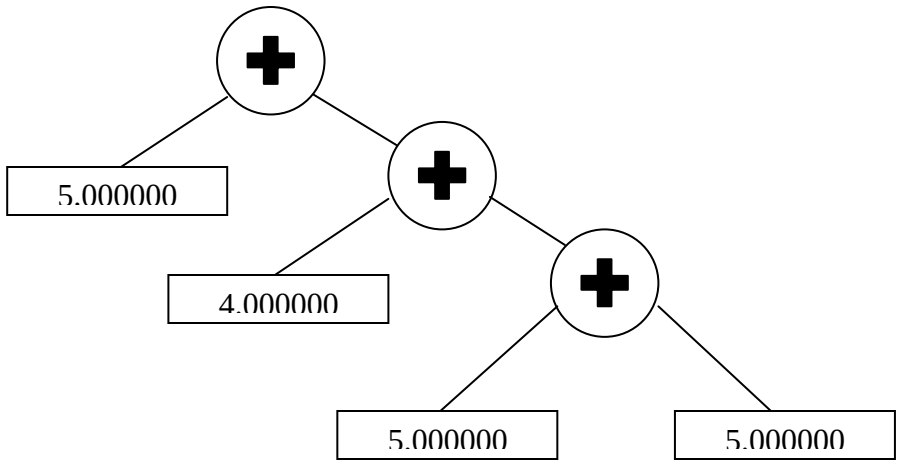


Diagram 6:
Sample output and parse trees

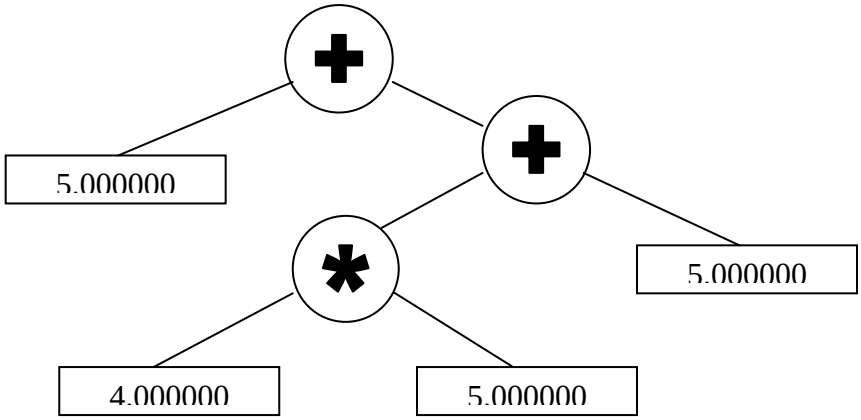
```
5+4+5+5;
==
Operator: +
   5.000000
   Operator: +
      4.000000
      Operator: +
         5.000000
         5.000000
```

```
5+4+5+5;
== 19.000000
```




```
5+4*5+5;
==
Operator: +
   5.000000
   Operator: +
      Operator: *
         4.000000
         5.000000
      5.000000
```

```
5+4*5+5;
== 30.000000
```




```
3/2+4(5-3^2)+2;
==
Operator: +
   Operator: /
      3.000000
      2.000000
   Operator: +
      Operator: *
         4.000000
         Operator: -
            5.000000
            Operator: ^
               3.000000
               2.000000
      2.000000
```

```
3/2+4(5-3^2)+2;
== -12.500000
```