CPSC 5520—Distributed Systems

Worksheet 1

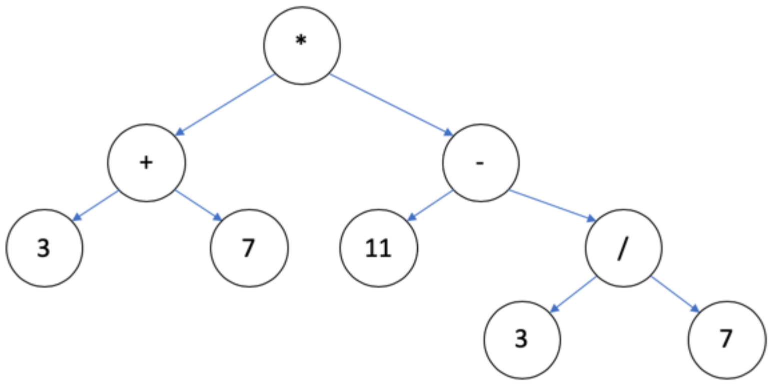
Decomposing a System into Tiers

Description

We start with a working monolithic system that we would like to turn into a distributed system of three tiers. The system takes a string mathematical expression and calculates an integer result. The input string might be something like:

```
( 3 + 7 ) * ( 11 – 2 / 2 )
```

and would, in that case, return the integer 100. This is done with a sequence of steps. First the string is parsed into an abstract syntax tree (AST):



and then the AST is evaluated to get the answer, 100. In our monolithic system it is encoded as shown in the attachment (also downloadable from Canvas). Basically, there are three methods: **`parse`**, **`evaluate`**, and **`calculate`** (which calls parse and evaluate).
We want to perform this as three tiers. Say I have a group that is really good at parsing and they are going to handle the parsing themselves, and then send off the AST to another group that is knowledgeable about how we want to perform the operations and they will do the evaluation of an AST. Meanwhile, we still have the original client running on its own node but will now farm out the work to the parsing group who in turn will subcontract the evaluation piece to the eval group. So, we want to end up with these three tiers:

1. Client who gets the string expression from the user and presents the answer.
2. Parser group provides a calculate service to the first tier (takes a string; returns an integer). Parser group parses the expression and sends the evaluation out to tier 3 whose result is then passed back to tier 1.
3. Calculate service—takes an AST and returns an integer.

Decomposition

Describe how you would go about taking the original monolithic code and distributing into three tiers. Starting from three copies of **`calculator.py`** copied into **`tier1.py`**, **`tier2.py`**, and **`tier3.py`**, explain what code would be removed and added to each tier.

Write the bits of code that are added to each tier. Try doing it pen-on-paper as much as possible without looking at examples. Then, actually perform the experiment on your computer.