Assignment 6 (Optional)

Exercise 1 (LLVM Code Generation, 10 Bonus Points)
Based on the simple LLVM example program (fibonacci.cpp, see our class website), create an LLVM-based "subscript"-compiler for simple binary arithmetic expressions. Example: \((v_0 + v_1) \times (v_2 - v_3)\), in tree representation:

\[
\begin{array}{c}
\times \\
+ \\
\quad v_0 \\
\downarrow \\
- \\
\quad v_1 \\
\downarrow \\
\quad v_2 \\
\downarrow \\
\quad v_3
\end{array}
\]

Write a function similar to CreateFibFunction that takes an llvm::Module (like fibonacci.cpp), an llvm::LLVMContext (like fibonacci.cpp) and the root of a binary arithmetic expression tree as parameters. The tree’s nodes contain the operators \(\times, \div, +, -\), constants and variables \((v_0, \ldots, v_{n-1})\).

The function generates a function \(f\) that can be called with \(n\) integer arguments and returns the result of the expression, e.g. in the example above \(f(1, 2, 5, 2)\) returns 9 (since \((1 + 2) \times (5 - 2) = 9)\).

Your code may be completely independent from your database system’s code base.