An Add-Only Calculator

12 + 57 + 8 + 10 =
S = 0
do {
    get next token
    if (token is not a number) error
    add token to S
    get next token
} while (token is "+")
if (token is not ") error
return S
Adding and Multiplying

\[12 + 57 \times 8 + 10 \times 5 \times 3 + 2 = \]
Adding and Multiplying

12 + 57 * 8 + 10 * 5 * 3 + 2 =

```plaintext
S = 0
do {
P = 1
do {
    get next token
    if (token is not a number) error
    multiply P by token
    get next token
} while (token is "*")
add P to S
} while (token is +)
if (token is not "+") error
return S
```
Parentheses

\[ 12 + 57 \times (8 + 3 \times 2) + 10 \times 5 \times 3 + 2 = \]
Parentheses

12 + 57 * (8 + 3 * 2) + 10 * 5 * 3 + 2 =

```c
int expr() {
    S = sop()
    if (token is not "=") error
    return S
}
int sop() {
    S = 0
    do {
        P = 1
        do {
            get next token
            if (token is "(") {
                N = sop()
                if (token is not ")") error
            } else if (token is a number)
                N = token
            else if (token is not a number) error
            multiply P by N
            get next token
        } while (token is ")")
        add P to S
    } while (token is +)
    return S
}
```
Context-Free Grammars

\[
\begin{align*}
  sum & \rightarrow number \\
  sum & \rightarrow sum + number 
\end{align*}
\]
Context-Free Grammars

\[
\begin{align*}
  sum & \rightarrow \ number \\
  sum & \rightarrow \ sum + number \\
  sum & \rightarrow \ product \\
  sum & \rightarrow \ sum + product \\
  product & \rightarrow \ number \\
  product & \rightarrow \ product \ast number
\end{align*}
\]
Context-Free Grammars

sum → number
sum → sum + number

sum → product
sum → sum + product

product → number
product → product * number

sum → product
sum → sum + product

product → term
product → product * term

term → number
term → ( sum )