0. Here is a context-free grammar for our calculator language. The start symbol is S. Intuitively, S generates statements, E expressions, O operators, F floating-point constants, and I integer constants. V generates all variables — that is, strings of non-operator, non-parenthesis symbols that cannot be interpreted as numbers, either because they are a single period, have too many periods, or have non-digit characters among the periods.

$$
\begin{aligned}
S &\rightarrow V = E \mid E \\
E &\rightarrow (E) \mid EOE \mid V \mid F \mid I \\
O &\rightarrow + \mid * \mid \hat{} \mid - \mid / \\
F &\rightarrow .I \mid I. \mid I.I \\
I &\rightarrow D \mid DI \\
D &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
V &\rightarrow . \mid B.B.B \mid BCB \\
B &\rightarrow \epsilon \mid .B \mid DB \mid CB \\
C &\rightarrow \text{any terminal symbol other than } +, *, \hat{}, -, /, (,), ., \text{ and the ten digits}
\end{aligned}
$$

In the grammar above I have ignored white space altogether. We can account for white space by altering a few productions and introducing nonterminals to generate strings of white space symbols.

$$
\begin{aligned}
S &\rightarrow ZVZ = ZEZ \mid ZEZ \\
E &\rightarrow Z(ZEZ)Z \mid ZEYOYEZ \mid ZVZ \mid ZFZ \\
C &\rightarrow \text{any terminal other than } +, *, \hat{}, -, /, (,), ., \text{ the digits, and white space} \\
Z &\rightarrow \epsilon \mid Y \\
Y &\rightarrow W \mid WY \\
W &\rightarrow \text{any white space symbol}
\end{aligned}
$$

1. Let A be a regular language over Σ. Then there exists a regular expression α that matches A (and nothing else). We wish to produce a context-free grammar G_α that generates the same language A. We proceed by structural induction on α. The base cases are easy; I'll leave them to you. There are three inductive cases: $\alpha = \beta + \gamma$, $\alpha = \beta\gamma$, and $\alpha = \beta^*$.

Assume that $\alpha = \beta + \gamma$ and that G_β and G_γ are CFGs that describe the languages corresponding to β and γ. (This is the inductive hypothesis.) There is a standard procedure for producing a CFG that generates the union $L(G_\beta) \cup L(G_\gamma)$. (This was done on homework; I'm happy to fill in details in person.) This union CFG generates the language of $\alpha = \beta + \gamma$.

Similarly, assume that $\alpha = \beta\gamma$; there is a standard procedure, done on homework, for constructing a CFG for α from those for β and γ.

Finally, assume that $\alpha = \beta^*$, where G_β is a CFG that generates the language of β. Let S_β be the start symbol of G_β. Let S_α be any symbol that does not occur in the nonterminal and terminal alphabets of G_β. Construct a CFG by adding to G_β the production

$$S_\alpha \to \epsilon \mid S_\beta S_\alpha;$$

the start symbol of this CFG is S_α. Then this new CFG generates the language of α.

We have shown that, no matter how the regular expression α is constructed, there is always a CFG that generates the same language. Therefore any regular language is also context-free.

2. We will show that the language A consisting of all strings of the form $x = y + z$, where $x, y, z \in \{0,1\}^*$ and $x = y + z$ is a correct equation of base-2 numbers, is not context-free, using the pumping lemma. Let $k \geq 0$ be given. Let z be the string

$$1^k 0 = 1^k + 1^k,$$

which is in A and of length at least k. Suppose that z is subdivided as $z = uvwxy$, where $|vwx| \leq k$ and $vx \neq \epsilon$. We will show that uv^2wx^2y cannot be in A. There are several cases to consider. Most are easy; we leave the difficult case for the end.

If vwx is entirely contained in the left-hand side of the equation — that is, in $1^k 0$ — then uv^2wx^2y is an equation where the left-hand side has a different numerical value than that of $1^k 0$, but the right-hand side still has numerical value equalling that of $1^k 0$. Hence the equation is false and the string is not in A.

Suppose that vwx is entirely contained in the right-hand side of the equation. If either v or x contains $+$, then uv^2wx^2y contains more than one $+$ and hence is not in A. If vwx is entirely contained in the first 1^k-term on the right-hand side of the equation, then uv^2wx^2y is a similar equation with just this term changed; in fact, the numerical value of the term is changed, so the equation is false and not in A. Similarly, if vwx is entirely contained in the second 1^k-term, then uv^2wx^2y is not in A. The final subcase occurs when v is contained in the first 1^k-term and x is contained in the second. In this subcase the right-hand side of uv^2wx^2y has greater numerical value than the right-hand side of $uvwxy$, and so uv^2wx^2y cannot be in A.

The only remaining cases occur when vwx intersects both the left- and right-hand sides of the equation $uvwxy$. Since $|vwx| \leq k$ this forces $k \geq 3$. If either v or x contains $=$, then uv^2wx^2y contains more than one $=$ and hence is not in A. Now the only remaining case occurs when v is contained in the left-hand side and x in the right-hand side. Because $|vwx| \leq k$, it must be that x is contained in the first 1^k-term on the right-hand side. Thus $x = 1^\ell$ for some $1 \leq \ell \leq k$. Consider uv^2wx^2y. This is an equation with right-hand side $1^{\ell+k} + 1^k$. Let's do some arithmetic with base-2 numerals:

$$\begin{aligned} 1^{\ell+k} + 1^k &= 1^\ell 0^k + 1^k + 1^k \\ &= 1^\ell 0^k + 1^k 0 \\ &= 1^\ell 0^k + 11^{k-1}0 \\ &= (1^\ell + 1)1^{k-1}0 \\ &= 10^\ell 1^{k-1}0. \end{aligned}$$

For the equation uv^2wx^2y to be true, its left-hand side must be $10^\ell 1^{k-1}0$, which contains the 0-symbol multiple times. We deduce that v contains 0, that $v = 1^m 0$ for some $0 \leq m \leq k$, and that the left-hand side of uv^2wx^2y must be $uv^2 = 1^{k-m}1^m 0 1^m 0$. This can equal $10^\ell 1^{k-1}0$ only if $m = k - 1$ and $k = 1$. But $k \geq 3$. Thus there is no way for uv^2wx^2y to be in A.

In all cases we have shown that there is no way to subdivide z as $uvwxy$ such that uv^2wx^2y is in A. Therefore, by the pumping lemma, A is not context-free.

3. Intuitively, a PDA on each step of its computation reads in zero or one input symbol, pops its stack, uses the resulting information to transition to a new state, and pushes zero or more stack symbols. Intuitively, a two-stack PDA on each step of its computation reads in zero or one input symbol, pops from two stacks, uses the resulting information to transition to a new state, and pushes zero or more stack symbols onto each stack. There is no reason to assume that the two stack alphabets are identical; let's not. Formally, a two-stack PDA is a 9-tuple

$$M = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, \perp_1, \perp_2, s, F),$$

where Q is a finite set of states, Σ is a finite input alphabet, Γ_1 and Γ_2 are finite stack alphabets, $\perp_i \in \Gamma_i$ are the starting symbols for the two stacks, $s \in Q$ is the start state, $F \subseteq Q$ is the set of final states, and

$$\delta \subseteq (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma_1 \times \Gamma_2) \times (Q \times \Gamma_1^* \times \Gamma_2^*)$$

is the nondeterministic transition relation. M accepts by final state in the familiar way.

For any two context-free languages L_1 and L_2 over the same alphabet Σ we can construct a two-stack PDA M to accept $L_1 \cap L_2$ as follows. Let M_1 and M_2 be PDAs that accept L_1 and L_2, respectively, by final state. For the two-stack PDA M, declare $Q = Q_1 \times Q_2$, $s = (s_1, s_2)$, and $F = F_1 \times F_2$. Take M's Γ_i and $\perp_i$ straight from the M_i. For all transitions $((p_1, a, A_1), (q_1, \alpha_1))$ in M_1 and $((p_2, a, A_2), (q_2, \alpha_2))$ in M_2 (notice that the same input symbol a appears in both), construct a transition

$$(((p_1, p_2), a, A_1, A_2), ((q_1, q_2), \alpha_1, \alpha_2)$$

for M.

Now M is able, on input $x \in \Sigma^*$, to transition to a state $(q_1, q_2) \in Q$ if and only if on input x M_1 can transition to q_1 and M_2 can transition to q_2. Thus x can send M into a final state if and only if it can send M_1 into a final state and M_2 into a final state. Thus $L(M) = L(M_1) \cap L(M_2) = L_1 \cap L_2$.

From our homework we know that context-free languages are not closed under intersection. For example, $L_1 = \{a^n b^n c^m : n, m \geq 0\}$ and $L_2 = \{a^n b^m c^m : n, m \geq 0\}$ are context-free, but their intersection $L_1 \cap L_2 = \{a^n b^n c^n : n \geq 0\}$ is not. The construction above gives a two-stack PDA to accept $L_1 \cap L_2$. Therefore there is at least one language acceptable by two-stack PDAs that is not acceptable by one-stack PDAs.

It is easy to see that for any context-free language there exists a two-stack PDA to accept it; one can just take a one-stack PDA that accepts the language by final state and alter every transition so that it pops and pushes some symbol $\perp_2$ on its second stack. Therefore the set of languages accepted by two-stack PDAs is a strict superset of the set of context-free languages.