Notes for Monday, May 17th
(Notes supplementary to slides) Construction of a CFG to PDA follows this construction:

```
q0 \(\varepsilon, \varepsilon \rightarrow S\)
q1 \(\varepsilon, \varepsilon \rightarrow S\)
q2 \(\varepsilon, A \rightarrow w\)
q_{ACC}
```

In PDA to CFG proof, we can get the PDA into the required form by splitting things of the form

```
\(a, b \rightarrow c\)
```

Into

```
\(a, b \rightarrow \varepsilon\)
\(\varepsilon, \varepsilon \rightarrow c\)
```

and similarly

```
\(\varepsilon, \varepsilon \rightarrow \varepsilon\)
\(\varepsilon, c \rightarrow \varepsilon\)
```

(diverging from the slide discussion now)

Consider the example \(L = \{0^n1^n0^n | n \geq 0\}\). Can we make a CFG or PDA for \(L\)? If we had two stacks it would be easy. The attempted grammar

\[
S \rightarrow 0A1B0|\varepsilon \\
A \rightarrow 0A1|\varepsilon \\
B \rightarrow 1B0|\varepsilon
\]

doesn’t work. It turns out that \(L\) is not a CFL. To prove this, we’d need a pumping lemma for CFLs!

An example of how this pumping lemma would work: \(L = \{0^n1^n | n \geq 0\}\). We have already discussed the grammar for this in class:

\[
S \rightarrow 0S1|\varepsilon
\]

It has one variable, \(S\). So, \(|V| = 1\). A parse tree of height 2 would have the longest path have three nodes (with a leaf as a terminal), so there are two variables in the path. Since there is only one variable on \(V\), the path will repeat a variable due to pigeonhole. Hence, we can change the size of the tree!

For the string \(w = 01\), the tree is

```
S
  |  1
0  S  1
|  \varepsilon
```

We may replace the bottom \(S\) by the top \(S\) and “pump up” to get the string \(0^21^2\)...
...or again for $0^31^3$:

```
S
/ \  /
0 S 1
/  /
S 1
/  /
S 1
```

and so on. We may also “pump down” by replacing the upper $S$ by the lower $S$ to get the string $0^01^0 = \varepsilon$:

```
S
/ |  /
S 1
/ |  /
S 1
```

This outlines how the pumping lemma for CFLs works. Next time, a formal proof is given.