FROM CONTEXT-FREE GRAMMAR (CFG) TO NON-DETERMINISTIC PDA

General idea: first, we push a dollar sign $ into the stack of stacks, to check if a stack is empty. If $ is on top, there is nothing popped off. No symbol.

Reminder: in this formalization we do not have an easy way of pushing there is nothing popped off.
Second step (common for all CFGs):
we push the starting variable $S$ into the stack on top of $\$. 

\[
\text{start} \xrightarrow{\epsilon, \epsilon \rightarrow \$} \text{empty stack} \xrightarrow{\epsilon, \epsilon \rightarrow S} \text{Working state}
\]

The last state will be the working state
3.

* Now we need to take the grammar into account. Let's use a simple example: \( S \rightarrow aBb, \ B \rightarrow bSa, \ S \rightarrow \epsilon \)

* First, for each terminal symbol, we add a rule \( \alpha, \alpha \rightarrow \epsilon \).

* In this case, we have two terminal symbols: \( a \) and \( b \).

* So, we add two such rules:

\[
\begin{align*}
\alpha, \alpha & \rightarrow \epsilon \\
\beta, \beta & \rightarrow \epsilon 
\end{align*}
\]

What it means:

* If we see 'a' and there is 'a' on top of the stack, we pop it.

* If we see 'b' and there is 'b' on top of the stack, we pop it.
* Also, for each rule, $A \rightarrow u$, we pop $A$ and we push $u$ instead.
* How do we push $u = u_1 \ldots u_k$? It is a stack, so we need to push them one-by-one, from last to first:

\[ \varepsilon, A \rightarrow u_k \rightarrow \varepsilon, \varepsilon \rightarrow u_{k-1} \rightarrow \ldots \rightarrow \varepsilon, \varepsilon \rightarrow u_1 \]

* For example, to represent the rule $S \rightarrow aBb$, we first pop $S$, push $b$, then push $B$, then push $a$:

\[ \varepsilon, S \rightarrow b \rightarrow \varepsilon, \varepsilon \rightarrow B \rightarrow \varepsilon, \varepsilon \rightarrow a \]

* In the stack, they are in right order:
We do this for all the rules. Finally, we pop $ if it is on top of the stack. We did it in all previous examples. Grammar:

$ \rightarrow aBB$

$B \rightarrow bSa$, $S \rightarrow \varepsilon$, $a, a \rightarrow \varepsilon$, $b, b \rightarrow \varepsilon$

This is the desired pushdown automaton.
How do we translate derivation in CFG into acceptance by PDA?

Step by step

Example:

First, we apply the rule $S \rightarrow aBb$

Then, we apply the rule $B \rightarrow bSA$

Finally, we apply the rule $S \rightarrow \varepsilon$

We get the word $abab$

How do we accept this word by PDA?

**Start**

We start at starting state with empty stack

Then we push $\$\$

Then we push $S$, we are in working state
Now we used rule $S \rightarrow AB$.

Now we see the first symbol $a$, so we use rule $a \rightarrow \delta$. We write the PDA transitions.

Next, in our derivation, we use the rule $B \rightarrow \delta B\delta$. So we follow the PDA transitions and replace $S$ with a $B\delta$. We are back in working state.

Now we see symbol $a$, so we use the rule $a \rightarrow \delta$. We write the PDA state transitions.
On top of the stack is $b$, we see $b$ as the next unread symbol in our word $abab$ ("a" was already read), so we pop $b$.

Now we use the rule $S \rightarrow \epsilon$ to this rule corresponds transition $\epsilon, S \rightarrow \epsilon$.

We see $a$ on top of the stack, $a$ is the next unread symbol so we pop!
\begin{itemize}
    \item Now we see $b$ on top of the stack, $b$ is the next unread symbol $\text{\#}bab$, so already read.
    \item Now we only have $\$$ in the stack, so we pop it and go to final state.
    \item Now we are in the final state with empty stack.
    \item This means that the word $\text{abab}$ is accepted!
\end{itemize}
Summarizing, we want to accept the word abab, which was generated by the following sequence of rules:

\[ S \rightarrow aBb, \ B \rightarrow bSa, \ S \rightarrow \varepsilon; \]

\[ S \rightarrow aBb \]

Start

Initial phase

Final state

Done!
Tracing the derivation of a different word is the task.
Solution:

$ \Rightarrow aBb$

1st letter

$s \Rightarrow aBb$

2nd letter

$a, a \Rightarrow \epsilon$

3rd

$\Rightarrow abab$

4th

$\Rightarrow abab$

5th

$\Rightarrow abab$

6th

$\Rightarrow abab$

7th

$\Rightarrow abab$

8th

$\Rightarrow abab$

9th

$\Rightarrow \epsilon$

Final

We accepted the word ab ab ab ab ab ab ab
I started by:

- Defining symbols:
  - $S$: means $\text{sign}$, $P$ means $\text{positive integer}$, $D$ means digit ($0$ or $1$)

- Groupwork 2:
  - (to finish at home)

- I draw a PDA for a different grammar:
  - $S \rightarrow \epsilon$, $P \rightarrow D$
  - $S \rightarrow t$, $D \rightarrow 0, 1$

- Trace it on the example generated by this grammar:
  - $S \rightarrow t$
  - $P \rightarrow D$
  - $D \rightarrow 0$

- I trace word $+101$ generated.