**CFG to PDA conversion**

***Algorithm***

**Input** − A CFG, G = (V, T, P, S)

**Output** − Equivalent PDA, P = (Q, ∑, S, δ, q0, I, F)

**Step 1** − The PDA will have only one state {q}.

**Step 2** − The start symbol of CFG will be the start symbol in the PDA.

**Step 3** − All non-terminals of the CFG will be the stack symbols of the PDA and all the terminals of the CFG will be the input symbols of the PDA.

**Step 4** − For each production in the form **A → aX** where a is terminal and **A, X** are combination of terminal and non-terminals, make a transition **δ (q, a, A)**.

**Input /Output**

The program reads an example of Context Free Grammer from standard input and produce an equivalent push down automata on standard output. The states in the input consists of Terminals and Non terminals and Production rules.

* Input has been taken from file “input/input1.txt”
* Each subsequent line of file consists of all productions for a Non-Terminal.

**Sample Input 1 :**

S:aSa|bA

A:aS|bB

B:bB|b

**Sample Output 1 :**

*δ(q,ε,S) -> {(q,aSa),(q,bA)}*

*δ(q,ε,A) -> {(q,aS),(q,bB)}*

*δ(q,ε,B) -> {(q,bB),(q,b)}*

*δ(q,ε,B) -> {(q,bB),(q,b)}*

*δ(q,a,a) -> {(q,ε)}*

*δ(q,b,b) -> {(q,ε)}*

**Sample Input 2 :**

S:AC|CB

C:aCb|ε

A:aA|a

B:Bb|b

**Sample Output 2 :**

δ(q,ε,S) -> {(q,AC),(q,CB)}

δ(q,ε,C) -> {(q,aCb),(q,ε)}

δ(q,ε,A) -> {(q,aA),(q,a)}

δ(q,ε,B) -> {(q,Bb),(q,b)}

δ(q,ε,B) -> {(q,Bb),(q,b)}

δ(q,a,a) -> {(q,ε)}

δ(q,b,b) -> {(q,ε)}

**Explanation:**

* getInput() function will get the input from file and loads it in vector s.
* Pass each production rule to function cfg\_to\_pda() which will split the production rule into two parts 1. Non-terminal 2. Production rule
* Split the production rule with respect to | and makes transition function.