**PDA to CFG conversion**

***Algorithm***

**Step 1** − For every w, x, y, z ∈ Q, m ∈ S and a, b ∈ ∑, if δ (w, a, ε) contains (y, m) and (z, b, m) contains (x, ε), add the production rule Xwx → a Xyzb in grammar G.

**Step 2** − For every w, x, y, z ∈ Q, add the production rule Xwx → XwyXyx in grammar G.

**Step 3** − For w ∈ Q, add the production rule Xww → ε in grammar G.

**Input /Output**

The program reads an example of Push Down Automata from file and produce an equivalent grammar in output file. Input format is as follows.

* δ(state1,input,topStack) -> {(state2,newTopStack)}
* **Sample Input**

q 1 X q XX

q 1 Z q XZ

q 0 X p X

p 1 X p $

p 0 Z q Z

* **Sample Output**

[ qXp ] -> 1 [ qXp ] [ pXp ]

[ qXp ] -> 1 [ qXq ] [ qXp ]

[ qXq ] -> 1 [ qXp ] [ pXq ]

[ qXq ] -> 1 [ qXq ] [ qXq ]

[ qZp ] -> 1 [ qXp ] [ pZp ]

[ qZp ] -> 1 [ qXq ] [ qZp ]

[ qZq ] -> 1 [ qXp ] [ pZq ]

[ qZq ] -> 1 [ qXq ] [ qZq ]

[ qXp ] -> 0 [ pXp ]

[ qXq ] -> 0 [ pXq ]

[ pXp ] -> 1

[ pXq ] -> 1

[ pZp ] -> 0 [ qZp ]

[ pZq ] -> 0 [ qZq ]

[ pZp ] -> ε

[ pZq ] -> ε

[ qZp ] -> ε

[ qZq ] -> ε

* **Sample Input 2**

q a Z q XZ

q a X q XX

q ε Z p Z

q ε x p X

p b X p ε

p ε Z r ε

* **Sample Output 2**

**[ qZp ] -> a [ qXp ] [ pZp ]**

**[ qZp ] -> a [ qXq ] [ qZp ]**

**[ qZp ] -> a [ qXr ] [ rZp ]**

**[ qZq ] -> a [ qXp ] [ pZq ]**

**[ qZq ] -> a [ qXq ] [ qZq ]**

**[ qZq ] -> a [ qXr ] [ rZq ]**

**[ qZr ] -> a [ qXp ] [ pZr ]**

**[ qZr ] -> a [ qXq ] [ qZr ]**

**[ qZr ] -> a [ qXr ] [ rZr ]**

**[ qXp ] -> a [ qXp ] [ pXp ]**

**[ qXp ] -> a [ qXq ] [ qXp ]**

**[ qXp ] -> a [ qXr ] [ rXp ]**

**[ qXq ] -> a [ qXp ] [ pXq ]**

**[ qXq ] -> a [ qXq ] [ qXq ]**

**[ qXq ] -> a [ qXr ] [ rXq ]**

**[ qXr ] -> a [ qXp ] [ pXr ]**

**[ qXr ] -> a [ qXq ] [ qXr ]**

**[ qXr ] -> a [ qXr ] [ rXr ]**

**[ qZp ] -> ε [ pZp ]**

**[ qZq ] -> ε [ pZq ]**

**[ qZr ] -> ε [ pZr ]**

**[ qxp ] -> ε [ pXp ]**

**[ qxq ] -> ε [ pXq ]**

**[ qxr ] -> ε [ pXr ]**

**[ pXp ] -> b**

**[ pXq ] -> b**

**[ pXr ] -> b**

**[ pZp ] -> ε**

**[ pZq ] -> ε**

**[ pZr ] -> ε**

**[ pZp ] -> ε**

**[ pZq ] -> ε**

**[ pZr ] -> ε**

**[ qZp ] -> ε**

**[ qZq ] -> ε**

**[ qZr ] -> ε**

**[ rZp ] -> ε**

**[ rZq ] -> ε**

**[ rZr ] -> ε**

***FUNCTIONS & VARIABLES***:

delta -> stores symbol for delta(235)

dollar -> stores symbol for epsilon(238)

getInput() -> takes input from file and gets the current state,next state,input symbol,top of stack and new top of stack make\_PDA() -> creates a pda given the current state,next state,input symbol,top of stack and new top of stack

pda\_to\_cfg() -> generates cfg on the basis of the length of the new stack top(i.e 1 or 2)

pda\_to\_cfg\_states -> generates remaining cfg giving the epsilon($) productions .