CFG TO PDA CONVERSION

Theory of Computation Assignment

-Kapeel Suryavanshi (BT16CSE084)

Algorithm:

1. q will be the only state in my PDA.
2. For each production of the form A -> X, where X can be a combination of terminal and non-terminals, I will have a transition of the form $\delta(q,\varepsilon,A) = (q,X)$.
3. For each terminal symbol ‘a’ in my CFG, I will have a transition of the form $\delta(q,a,a) = (q,\varepsilon)$.

Implementation:

- **Variables and Data Type used:**

Data Type	Variable Name	Description
vector< pair<string,vector<string> > >	gram	Vector of pair of string and vector of string, storing the state and its production rules.
vector<string>	nonterm_states	Vector of string storing non-terminal states.
vector<string>	term_states	Vector of string storing terminal states.

- **Functions and their description:**
  - **int main():**
    Driver Function. It takes the input of the CFG from a file and calls CFG_to_PDA with appropriate parameters.
  - **void CFG_to_PDA(***vector<pair<string,vector<string> > > &gram, vector<string> &nonterm_states, vector<string> &term_states, string start_state***)**:
    This function does the main job of converting CFG to PDA. Actually, it only prints the PDA and does not actually convert it. Based on the rules given in the algorithm, the transition of the PDA corresponding to the production rule in CFG is printed.
- **Input:**
  - Input is take from file.
  - First line of input, starts with NT, depicting non-terminal states.
  - Second line of input starts with T, depicting terminal states.
  - Rest of the lines, are the production rules
  - Example :
    NT = S A B X Y
    T = a b c
    S -> AX | YB
    A -> aAb | ε
    B -> bBc | ε
    X -> cX | ε
    Y -> aY | ε

- **Output:**
  - ❖ First, the CFG given as input is printed.
  - ❖ Then, its corresponding PDA is printed.
  - ❖ Output of my program when above input was given is :