THEORY OF COMPUTATION
ASSIGNMENT

Q2.2 Given a PDA, convert it to CFG

BASIC STRUCTURE

1. **PDA_CFG.h** is the header file that contains all the variables and the function prototypes
2. **PDA_CFG.cpp** is where the conversion of PDA to CFG takes place, all the permutations are printed
3. **pda_cfg_main.cpp** is the driver function where the file input_pda.txt is read and the corresponding PDA is loaded onto the program which the PDA_CFG.cpp to convert PDA to CFG

VARIABLES

TYPE	NAME	DESCRIPTION
vector<string>	states	States in the PDA
vector<string>	input_symbols	Input symbol
vector<string>	init_state	Stores the initial state before transition
vector<string>	final_state	Stores the final state after transition
vector<string>	stack_top	Top of the stack
vector<string>	push_stack	Symbol to be pushed on the stack
ifstream	fin	Input Stream object to operate on file

FUNCTIONS

1. **FileRead() -** Reads a file input.txt which contains the grammar
   - Checks error in opening file.
   - Pushes the states in PDA present on Line 1 in states vector.
   - Pushes the initial state, input symbol and top of stack of LHS to corresponding vectors, trimming the first three characters δ(and last)
   - Pushes the final state and symbol to be pushed onto the stack of RHS to corresponding vectors trimming the parenthesis.

2. **DisplayPDA()** - Display the PDA read from the file

3. **Permute()** - The construction of an equivalent grammar uses variables each of which represents an “event” consisting of :
    1. The net popping of some symbol X from the stack, and
    2. A change in state from some p at the beginning to q when X has finally been replaced by ε on the stack

    Such a variable is represented by a composite symbol [pXq]

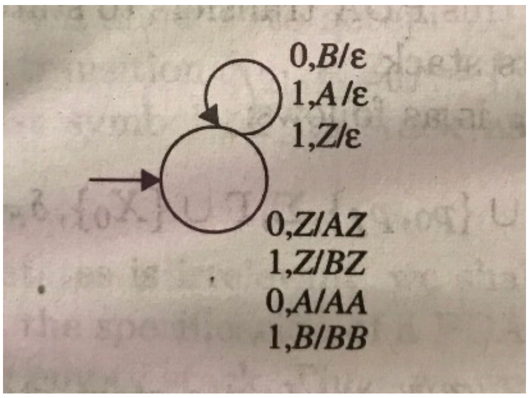
This function permutes all the variables in the Grammar (composite [pXq]) using standard recursive permutation technique.

4. **Print_Permute()** -Print all the Permutations found in Premute() function
5. **PDA_CFG()** - The PDA changes state as it pops stack symbols, so we note the state that it enters when it finally pops a level off its stack. The computation proceeds until each of the symbols on the stack is removed. All the variables generated from the states and stack symbols are printed

<u>INPUT</u>

input_pda.txt

```
q
δ(q,0,Z) = (q,AZ)
δ(q,1,Z) = (q,BZ)
δ(q,0,A) = (q,AA)
δ(q,0,B) = (q,ε)
δ(q,1,A) = (q,ε)
δ(q,1,B) = (q,BB)
δ(q,ε,Z) = (q,ε)
```



OUTPUT

```
PDA_TO_CFG — -bash — 100×30
~/Desktop/TOC/Q2/PDA_TO_CFG — -bash
MacBook:PDA_TO_CFG ayushsingh$ g++ pda_cfg_main.cpp
MacBook:PDA_TO_CFG ayushsingh$ ./a.out
-----------------------------------
THE INPUT PDA IS :
STATES = q
δ(q,0,Z) = (q,AZ)
δ(q,1,Z) = (q,BZ)
δ(q,0,A) = (q,AA)
δ(q,0,B) = (q,ε)
δ(q,1,A) = (q,ε)
δ(q,1,B) = (q,BB)
δ(q,ε,Z) = (q,ε)
-----------------------------------
Corresponding CFG :
START STATE = S -> [qXq]

δ(q,0,Z) = (q,AZ)
[qZq] -> 0 [qAq] [qZq]

δ(q,1,Z) = (q,BZ)
[qZq] -> 1 [qBq] [qZq]

δ(q,0,A) = (q,AA)
[qAq] -> 0 [qAq] [qAq]

δ(q,0,B) = (q,ε)
[qBq] -> 0

δ(q,1,A) = (q,ε)
[qAq] -> 1

δ(q,1,B) = (q,BB)
[qBq] -> 1 [qBq] [qBq]

δ(q,ε,Z) = (q,ε)
[qZq] -> ε

[qXq] -> ε
MacBook:PDA_TO_CFG ayushsingh$
```