|  |  |  |
| --- | --- | --- |
| **AIN SHAMS UNIVERSITY**  **FACULTY OF ENGINEERING**  **International Credit Hours Engineering Programs (i.CHEP)** |  | |
|  | | |
| Automata & Compatibility Project Documentation  **Prepared by Team 2**  **Submitted to**  **Prof. Dr. Gamal A. Ebrahim**  **T.A Eng. Sally E. Shaker** | | |
|  | |  |
|  | |

# Team Namelist:

|  |  |
| --- | --- |
| Name | ID |
| Abdulrahman Ayman Mahmoud | *19P6458* |
| *Ahmed Sameh Mohamed Mourad* | *19P5861* |
| Osama Ayman Mokhtar Amin | *19P1609* |
| *Amr Haithem Abd Almalek Shams* | *19P2352* |

**Table of Contents**

[Team Namelist: ii](#_Toc134539097)

[1. NFA TO DFA: 5](#_Toc134539098)

[2. CFG TO PDA: 6](#_Toc134539099)

[2.1 Conversion Steps 6](#_Toc134539100)

[2.2: Output Screenshots 7](#_Toc134539101)

**Table of Figures:**

[Figure 1: 8](file:///D:\University\Senior%201\Semester%208\Human%20Rights\Major%20Task\EL%20Basha%20Rights%20Final%20Ver.docx#_Toc133783595)

1. NFA TO DFA:

Text

# 2. CFG TO PDA:

## 2.1 Conversion Steps

1. **Inputting CFG Rules:** The program allows the user to input the context-free grammar rules in the following format: {S -> Aa, S -> c, A -> bc}, etc. This provides the necessary input to generate a pushdown automaton.
2. **Extracting The Start State**: The user has the option to input the starting state or leave it empty. If no starting state is specified, the program automatically selects the first encountered non-terminal in the input CFG rules as the starting state. This ensures that the pushdown automaton is generated correctly based on the provided rules.
3. **Extracting the Terminals & Non-terminals**: The program iterates through the input CFG rules and extracts the terminals, storing them in a list. It then performs the same operation to extract the non-terminals in the CFG rules.
4. **Adding the first two States**: The program adds two initial states to the pushdown automaton. The first state, qstart, is added as (ε, ε -> $), while the second state, q1, is added as (ε, ε -> S). These initial states serve as starting points for the pushdown automaton to begin processing the input strings.
5. **Adding Variable Productions to qloop:** The program adds all variable productions to the qloop state. It does this by popping the variable from the top of the stack and sequentially adding its corresponding productions. This process continues until all variable productions have been added to the qloop state.
6. **Adding Terminals to qloop:** The program adds all terminals to the qloop state. It does this by reading the terminal input from the user and popping it from the stack.
7. **Adding Accepting State (qaccept):** The program adds an accepting state, qaccept, to the pushdown automaton. It transitions to this newly added state by popping the $ symbol from the top of the stack and leaving the qloop state. This finalizes the process of generating the pushdown automaton from the input CFG rules.

## 2.2: Output Screenshots