FORMAL LANGUAGE AND AUTOMATA

REMOVAL OF EPLISON AND UNIT USELESS SYMBOLS

OMKAR SONAR(RA2112701010025)

N.HIRIDHARAN(RA2112701010010)

OBJECTIVE OF THE PROBLEM

- **Removal of Epsilon Productions:**
- **Identify Epsilon Productions:**
  - Look for productions of the form A → ε, where A is a non-terminal symbol.
  - Also, identify nullable non-terminals (non-terminals that can derive ε).
- **Remove Epsilon Productions:**
  - For each production A → ε, remove it.
  - For each production B → αAβ, add new productions without A, considering all possible combinations of A being present or absent in the derivation of α and β.

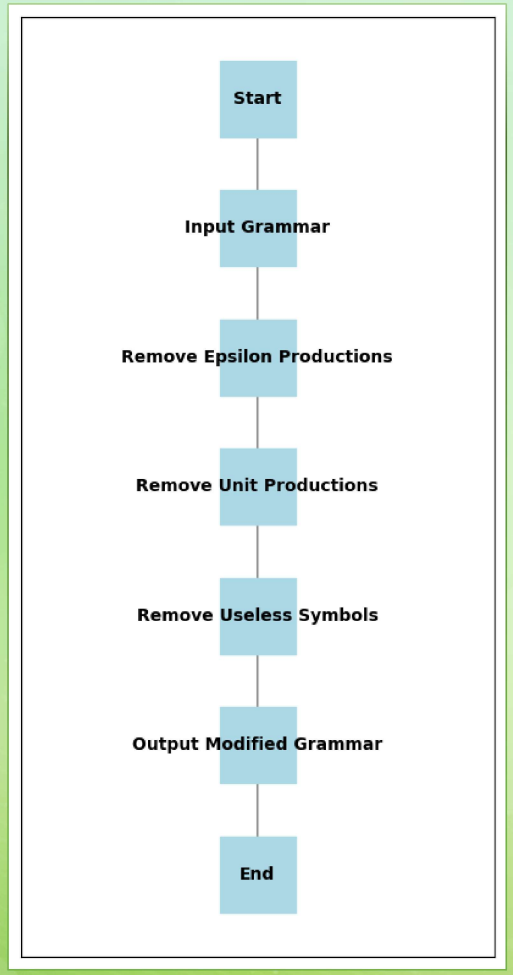
OBJECTIVE OF THE PROBLEM CTND.

- **Removal of Useless Symbols:**
- **Identify Non-Terminals and Terminals:**
  - Non-terminals that derive at least one string are called reachable non-terminals.
  - Terminals are always considered reachable.
- **Find Reachable Non-Terminals:**
  - Start with the start symbol and mark it as reachable.
  - Mark any non-terminal as reachable if it appears on the right-hand side of a production where all symbols are reachable.
- **Remove Unreachable Non-Terminals:**
  - Remove any non-terminals and their productions that are not marked as reachable.
- **Find Useful Terminals:**
  - Terminals that appear on the right-hand side of any production are useful.
- **Remove Useless Productions:**
  - Remove any production that involves non-terminals or terminals that are not useful.

APPLICATIONS

- Compiler Design
- Code Optimization
- Natural Language Processing (NLP)
- Automated Code Generation
- Error Detection and Reporting
- Educational Purposes
- Automated Testing

FLOWCHART:


Start
Input Grammar
Remove Epsilon Productions
Remove Unit Productions
Remove Useless Symbols
Output Modified Grammar
End

CONCLUSION:

- In conclusion, undertaking a project focused on the removal of epsilon unit productions and useless symbols in a context-free grammar serves as a valuable exploration into the optimization of language representations. By systematically eliminating these elements, the grammar becomes more concise and meaningful, facilitating efficient parsing, code generation, and error detection in various computational applications. This project not only provides practical insights into formal language theory but also underscores the importance of refining grammars for improved language processing and software development. Through this endeavor, one gains a deeper understanding of the intricate balance between grammar complexity and computational efficiency, contributing to foundational knowledge in fields such as compiler design, natural language processing, and automated code generation.

OUTPUT:

```
Original grammar:
Terminals: {'b', 'c', 'a'}
Non-terminals: {'A', 'S', 'B', 'C'}
Start symbol: S
Productions:
A -> aA |
S -> AB | C
B -> bB | c
C -> S | cC
Grammar after removing epsilon productions:
Terminals: {'b', 'c', 'a'}
Non-terminals: {'A', 'S', 'B', 'C'}
Start symbol: S
Productions:
A -> aA | a
S -> AB | B | C
B -> bB | c
C -> S | cC
Grammar after removing unit productions:
Terminals: {'b', 'c', 'a'}
Non-terminals: {'A', 'S', 'B', 'C'}
Start symbol: S
Productions:
A -> aA | a
S -> AB | bB | c | cC
B -> bB | c
C -> AB | cC
```