***<MAKE A COPY OF THIS TEMPLATE FOR EACH LAB>***

**Course - System Programming and Compiler Construction (SPCC)**

|  |  |
| --- | --- |
| **UID** | 2021300126 |
| **Name** | Pranay Singhvi |
| **Class and Batch** | TE Computer Engineering - Batch C |
| **Date** | 07-02-2024 |
| **Lab #** | 2 |
| **Aim** | To create a Parse Tree and compute the sets firstpos, lastpos, and followpos for each node in the Parse Tree for a given regular expression or grammar. |
| **Objective** | The objective is to construct a Parse Tree from a given regular expression, employing parsing techniques. Calculate firstpos, lastpos, and followpos sets for each node in the Parse Tree, enabling efficient analysis of positions where symbols can begin, end, and follow within the parsed expression, contributing to language parsing and analysis. |
| **Theory** | Theory:  1. Parse Tree Construction:  A Parse Tree is a hierarchical representation of the syntactic structure of a given expression or grammar. It visually illustrates how the input string is parsed according to the grammar rules. Recursive descent parsing or other algorithms can be employed to build a Parse Tree, where non-terminals are represented by internal nodes and terminals by leaves.  2. Firstpos Calculation:  The `firstpos` of a node in the Parse Tree is the set of positions in the input string where the node can be the first symbol of a valid parse. For terminal nodes, `firstpos` is simply the set containing the node's position. For non-terminals, it is the union of `firstpos` sets of its child nodes.  3. Lastpos Calculation:  The `lastpos` of a node is the set of positions where it can be the last symbol of a valid parse. For terminals, `lastpos` is the set containing the node's position. For non-terminals, it is the union of `lastpos` sets of its child nodes. The `lastpos` of a node is essential for handling concatenation.  4. Followpos Calculation:  The `followpos` of a non-terminal node represents the set of positions that can follow any occurrence of the non-terminal in a valid parse. This is determined by examining the `firstpos` sets of nodes following the non-terminal in the Parse Tree, considering concatenation and closure operations.  Procedure:  - Parse Tree Construction: Utilize recursive descent parsing or other parsing algorithms to construct a Parse Tree, adhering to grammar rules.  - Firstpos Calculation: Traverse the Parse Tree, calculating `firstpos` sets for each node based on its type (terminal or non-terminal) and the `firstpos` sets of its children.  - Lastpos Calculation: Traverse the Parse Tree (possibly in reverse), calculating `lastpos` sets for each node by considering its type and the `lastpos` sets of its children.  - Followpos Calculation: Use information from `firstpos` and `lastpos` to compute `followpos` for non-terminal nodes. Account for concatenation and closure operations.  Output:  - Display the constructed Parse Tree to visually represent the parsing structure.  - Provide the `firstpos`, `lastpos`, and `followpos` sets for each node, enabling further analysis of the parsing process.  Applications:  Efficient parsing and analysis of regular expressions and grammars, crucial in the fields of compiler construction, syntax analysis, and language processing, facilitating the generation of abstract syntax trees and subsequent code generation or interpretation. |
| **Implementation / Code** |  |
| **Output** |  |
| **Conclusion** |  |
| **References** |  |