National University of Computer and Emerging Sciences
Fast School of Computing Spring 2025

CS-4031-Compiler Construction
Assignment #03

Instructions

1. The following assignment must be completed in the same pairs as Assignment 1.
2. The assignment must be attempted in the C programming language.
3. Your code must be generic and reusable.
4. Usage of built-in functions or libraries related to compiler construction is not allowed.
5. Ensure proper indentation and comments in the code in addition to a presentable output of the program.
6. Plagiarism in any form (copying from others, copying from the internet, etc.) is strictly prohibited. If your work is found to be plagiarized, you will be awarded zero marks for the assignment.

Submission Guidelines

1. Combine all your work into one folder and name it as follows: `RollNumber1-RollNumber2-Section.zip` (e.g., `22i1234-22i5678-A.zip`).
2. Submit the .ZIP file on Google Classroom before the deadline. Submissions through other means (e.g. email) will not be accepted.

Assignment Overview

This assignment is a continuation of assignment 2. In this assignment, you are required to utilize the LL(1) parser designed and implemented in Assignment 2 to parse a sequence of space-separated terminal symbols (strings) provided in a text input file by implementing a parsing stack. The strings should contain space-separated terminal symbols to ensure that each terminal can be easily identified and parsed without the need for token generation.

Program Requirements

- Use the LL(1) parser created in Assignment 2.
- Read an input string from a file.
- Parse the string using the LL(1) parsing table by creating a parsing stack, showing the parsing steps and stack contents at each step.
- Display meaningful error messages for syntax errors and handle them gracefully.
- Print a clear success or failure message after parsing.
- Display the stack at each step.

0.1. Program Requirements

0.1.1 LL 1 parser Design

- utilize the parser designed in assignment 2.
- use assignment 2 as a pre-requisite to this assignment i.e. giving a CFG as input, generating first follow sets and a parse table for the CFG along with all the error handling methods as specified in assignment 2.

0.1.2 Input File

- The input file will contain one or more lines.
- Each line is a string consisting of space-separated terminals.
- These strings are to be parsed using your LL(1) parser.

0.1.3 LL 1 Parser integration

- Create a stack to simulate the LL(1) parser.
- The stack should be initialized with the start symbol and $ (end marker).

0.1.4 String Parsing

This step involves the utilization of stack for parsing the step. You are supposed to implement and display all the steps involved in parsing using a stack.

- Show the stack contents, current input symbol, and action taken at each step.
- If a terminal matches, pop and advance.
- If a non-terminal is on top of the stack, use the parsing table to expand it.
- If an error such as a missing entry or a mismatch occurs, handle the error gracefully as per the methods taught in class or described in the book. The minimum acceptable handling would be to detect and display the error encountered, followed by either skipping the error-causing entry or stopping the parsing completely.

0.1.5 Error Handling

- detect and display any errors like missing symbols, mismatched symbols or mismatched productions.

0.1.6 Output

- Stack's content at each parsing step.
- Parsing success/failure message
- error messages if encountered.

Example

0.1.7 Input

int x;
x = 5 + ;
if ($x > 0${
x = x - 1;
}

0.1.8 Output

Line 3: Syntax Error: Unexpected token ';' after '+'
Line 4: Syntax Error: Expected ')' before ''
Line 5: Parsing continued after error recovery.
Parsing completed with 2 errors.

Deliverables

1. Source code files in C.
2. A sample input file containing the input strings.
3. Output as specified above.
4. Parsing success or failure message
5. A brief report (in PDF format) describing your approach, any challenges faced, and how you verified the correctness of your program.

Good luck!