**Name: Hammad Shabbir, Iqrash Qureshi**

**ID: 22I-1140, 22I-1174**

**Section : CS-F**

**Complier Construction A4**

**JSON to Relational CSV Converter**

This project takes JSON input, parses it using Flex (scanner) and Yacc/Bison (parser), constructs an Abstract Syntax Tree (AST), and then generates one or more CSV files representing the structured data.

**Setup and Installation on Ubuntu**

**To install Flex and Yacc/Bison:**

**Install Flex:**

* sudo apt-get update
* sudo apt-get install flex

**Install Bison:**

* sudo apt-get install byacc
* sudo apt-get install bison
* sudo apt-get install bison++
* sudo apt-get install byacc-j

**Verify Installation:**

* flex --version
* yacc --version

**Project Structure:**

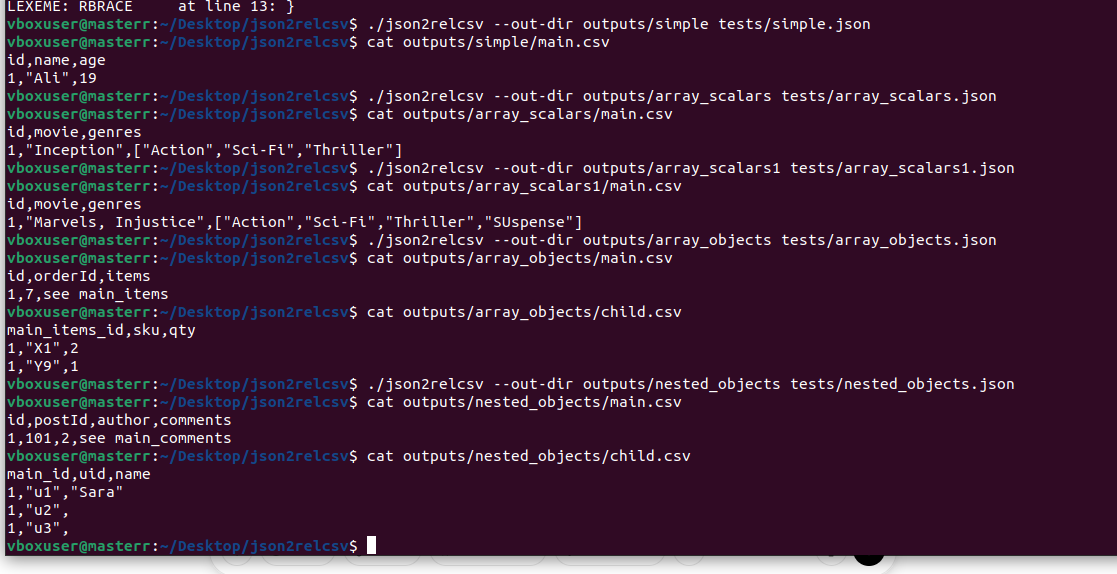
* `scanner.l`: Lexical analyzer for recognizing JSON tokens
* `parser.y` : YACC/Bison grammar file for parsing JSON structure
* `ast.h` : AST node structure and utility functions
* `tables.h` : Functions to transform AST into relational tables
* `Makefile`: Build configuration to compile the entire project
* `tests/` : Folder containing test JSON input files
* `outputs/` : Output directory for generated CSV files
* `json2relcsv`: Final compiled binary for JSON to CSV transformation

**Compile Project**

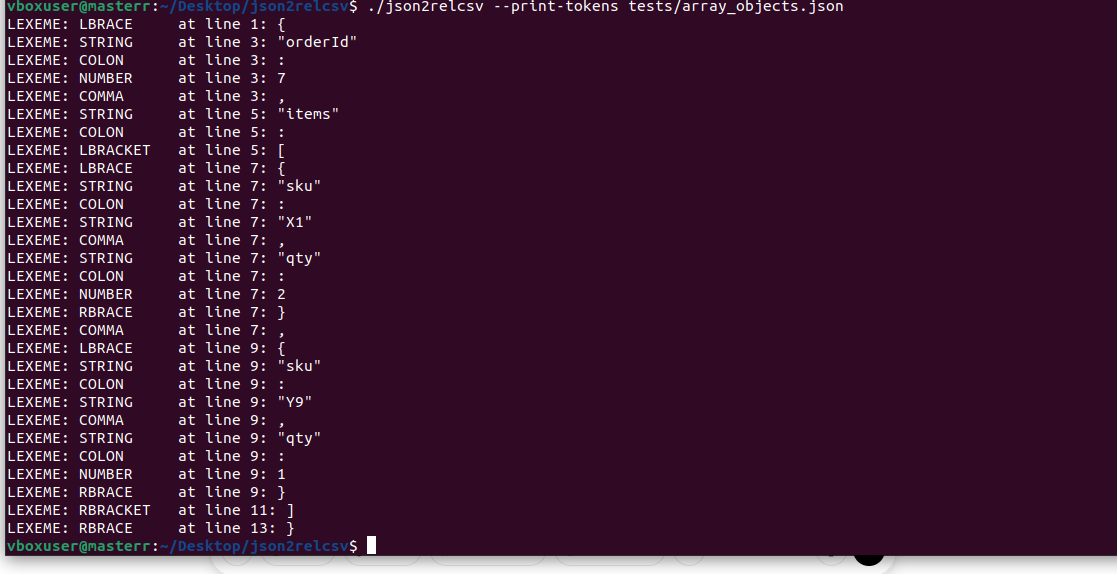
* make

**How to USE!**

**Generate CSV Output from JSON:**

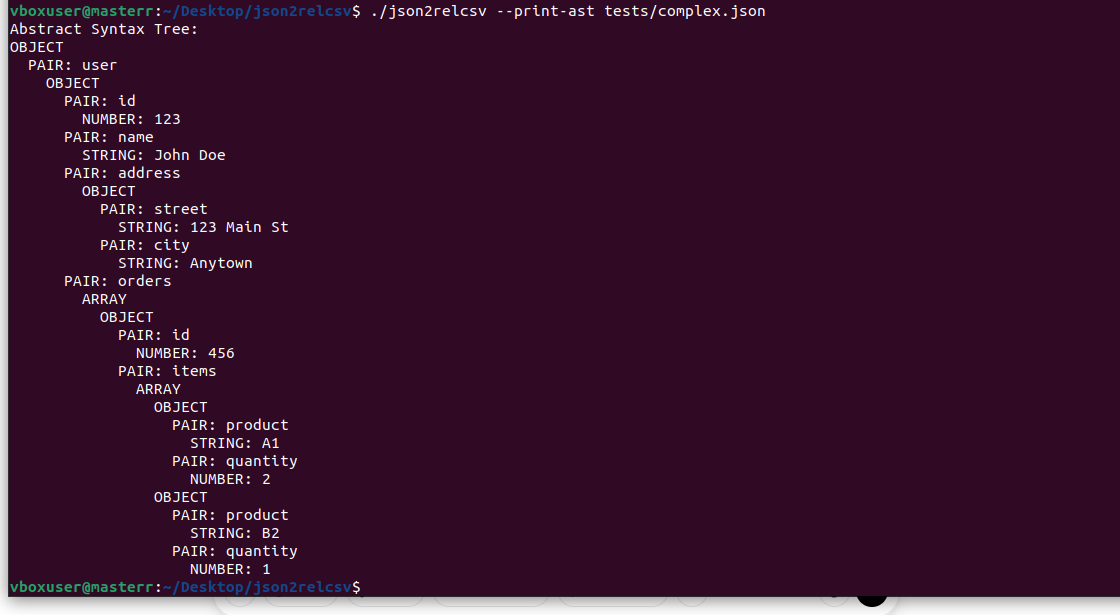
* ./json2relcsv --out-dir outputs/array\_scalars tests/array\_scalars.json
* cat outputs/array\_scalars/main.csv
* 

**Print Tokens:**

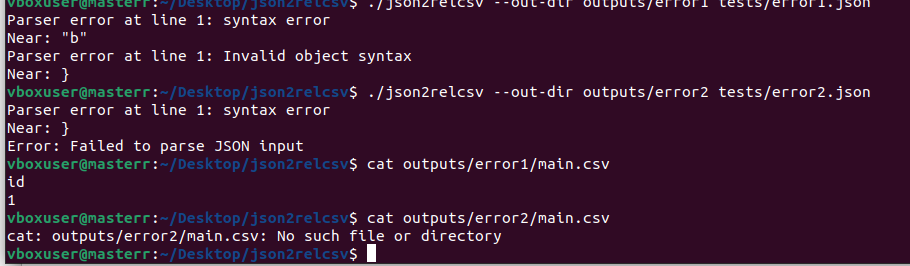
* ./json2relcsv --print-tokens tests/array\_objects.json
* 

**Print AST tree**

* ./json2relcsv --print-ast tests/complex.json



**Print Error**



**Theory & Approach (Key Concepts)**

**1. Scanner (Flex - scanner.l)**

* Identifies lexemes like {, }, [, ], :, ,, string, number, etc.
* Converts source input into tokens passed to the parser

**2. Parser (Yacc/Bison - parser.y)**

* Defines grammar rules for JSON
* Builds an Abstract Syntax Tree (AST) during parsing

**3. AST (ast.h)**

* Custom tree structure representing the JSON data hierarchy
* Nodes for objects, arrays, literals, etc.

**4. Relational Table Generator (tables.h)**

* Walks the AST and transforms hierarchical JSON into flat relational CSV format
* Each JSON object/array corresponds to a table

**5. Makefile**

* Automates the build process (flex + yacc + compilation)