# Recreating the Lisa Pascal Compiler Using Flex and Bison

## Introduction

For our final project, we were tasked with recreating the Lisa Pascal Compiler using Flex and Bison based on a given user manual. This project involved understanding the workings of lexical analysis and syntax parsing, constructing an effective symbol table, and ensuring that our Pascal compiler would successfully parse and analyze Pascal programs. This essay discusses the project's implementation, challenges encountered, and the learning outcomes from this experience.

## Project Description

The Lisa Pascal Compiler project aimed to create a compiler capable of lexical analysis, syntax analysis, and semantic analysis for Pascal programs. Flex was utilized to generate the lexical analyzer, which tokenizes the input source code. Bison was used to build the parser, applying grammar rules to the sequence of tokens produced by the lexical analyzer.

### Lexer Specification

The lexical analyzer specification, defined in 'lisa.l', includes tokens for Pascal keywords, operators, identifiers, and numerical values. Special attention was given to handling comments and white spaces. Multi-line comments were supported using state handling, and the line number was tracked for better error reporting.

### Parser Specification

In 'lisa.y', the parser's grammar rules govern the structure of valid Pascal programs. It defined token types and precedence rules and utilized semantic actions to populate the symbol table with correctly parsed identifiers. The parser's rules covered various Pascal constructs such as program headers, block structures, and statement lists.

### Symbol Table Implementation

A crucial component of the compiler was the symbol table, implemented as a hashmap. This table stored information about identifiers, including their names, scope (or ambitos), type, and the line number on which they were declared or referenced. The symbol table allowed for quick lookups to check if identifiers were already declared, ensuring semantic correctness.

## Project Challenges

Throughout the project, we encountered several challenges, including:

- Understanding Flex and Bison: Gaining proficiency in these tools required a considerable amount of time to understand their syntax and capabilities, especially since they are not commonly used in everyday programming tasks.

- Error Handling: Implementing comprehensive error handling was vital to ensure the compiler provided meaningful feedback for both syntax and semantic errors. Accurately tracking line numbers and managing nested scopes were key aspects we had to address.

-Memory Management: Managing dynamic memory allocation for the symbol table entries and ensuring no memory leaks or segmentation faults occurred was a substantial technical challenge.

## Learning Outcomes

This project provided valuable lessons and skills, including:

- Compiler Construction: We gained practical experience in building compilers, learning how lexical analyzers and parsers work in tandem to process source code.

- Flex and Bison Proficiency: Working with Flex and Bison improved our understanding of these powerful tools used for lexical and syntax analysis.

- Problem-Solving Skills: The challenges faced during the project honed our problem-solving abilities, particularly in debugging complex issues involving compiler design.

- Team Collaboration: Collaborating as a team, we learned effective communication and project management techniques to divide tasks and integrate our work seamlessly.

## Conclusion

Recreating the Lisa Pascal Compiler was a challenging yet rewarding experience. It provided deep insights into the intricacies of compiler design and enhanced our technical capabilities in using Flex and Bison. The project also underscored the importance of thorough testing and robust error handling in software development. Overall, this project has equipped us with practical skills and knowledge that will be beneficial in our future endeavors in computer science and software engineering.