

**Department of Computer Science and Engineering**

| **Course Code: CSE 420** | **Credits: 1.5** |
| --- | --- |
|  |  |
| **Course Name: Compiler Design** | **Semester:** |
|  | **Fall 2023** |
|  |  |

**1 Introduction**

In the last assignment, we have constructed a sample of syntax analysis via matching the grammar with the tokens which came from lexical analyzer stages.

In this assignment, we will construct a parse tree, basically the last part of the syntax analysis of a compiler for a subset of the C language. That means we will perform syntax analysis with a grammar rule generating the parse tree then print the terminal nodes of the parse tree if it is syntactically correct.

**2 Language**

Our chosen subset of the C language has the following characteristics:

* There can be multiple functions. No two functions will have the same name.
* There will be no pre-processing directives like #include or #define.
* Variables can be declared at suitable places inside a function. Variables can also be declared in the global scope.
* The TreeNode class which is designed to represent nodes in a tree structure. It provides methods for creating nodes, managing child nodes, and traversing the tree structure in a post-traversal order.

**3 Tasks**

You have to generate a parse tree and print all the terminal nodes of the parse tree if they are syntactically correct according to the Syntax Analysis stages

**3.1 Tree generation using Syntax Analysis**

For the syntax analysis part you have to do the following tasks:

* You are given a modified lex file named **“lex\_analyzer.l”** to use it with your Yacc file. Try to understand the syntax and tokens used.
* The ***symbol\_info.h*** file which contains the symbol\_info class is designed to store information about symbols in a language, including their names, types, and associated syntax tree nodes.
* The ***TreeNode.h*** file contains the TreeNode class which is designed to represent nodes in a tree structure. It provides methods for creating nodes, managing child nodes, and traversing the tree structure in a post-traversal order.
* The ***syntax\_analyzer.y*** file is the bison file where the necessary grammars for a C function and the rules for those grammars are described to generate a syntax tree from ***input.txt*** file, and logs the tree structure to an output file named ***my\_log.txt*** file. On the main method of this file, there is a ***yyparse()*** function which is responsible for the parsing and construction of the syntax tree. Here the rules will be given and action should be implemented.

**4 Input**

The input will be a text file containing a c source program.File name will be given from the command line. Sample input and sample output are given in the txt file.

**5 Output**

In this assignment, there will be one output file. The output file should be named as <Your\_student\_ID>\_log.txt. This will contain the terminal nodes generated from the parse tree if the syntax is correct. For a syntax error output file it will give an error message

For more clarification about input-output check the supplied sample I/O files given in the lab folder. You are highly encouraged to produce output exactly like the sample one.

**6 Submission**

1. In your local machine create a new folder whose **name is your student id.**
2. Put the lex file named as **<your\_student\_id>.l**, the Yacc file **<your\_student\_id>.y** and a script named **script.sh also the input file** (modifying with your own filenames), **and the output file** in a folder **named with your student id**. **DO NOT** put any generated lex.yy.c file or any executablefile in this folder.
3. Compress the folder in a **.zip file** which should be **named as your student id**.
4. Submit the .zip file.

**Failure to follow these instructions will result in a penalty.**

**Bonuns Task:**

**Print the input file as like the input file one maintaining all the formats. For Example:**

**Sample input :**

**int square(int y) {**

**return y \* y;**

**}**

**Sample output 👍**

**Reference**

**To have a better understanding of Lab 03 , please go through Section 5.1, 5.2 and 5.3.1 from the reference compiler textbook 'Compilers: principles, techniques, and tools' written by Aho, Lam. Sethi. and Ullman.**