Daffodil International University

Department of Computer Science and Engineering

Lab Manual

Version: 2018.01

Course Code: CSE 332
Course Title: Compiler Design Lab

Table of Contents

Session 1: Introduction

Intended Learning Outcome:
a. Make a simple calculator
b. Testing programming basic of the students.

Expected skills:
a. Solving various simple computing problems

Tools Required:
a. Text editor and C/C++ Compiler
b. IDE such as CodeBlocks

Session Detail:

1. Read something from user
2. Calculate using various operators such as addition, subtraction etc.
3. Input/output arrays of different size
4. Reverse an array
5. Calculate the summation and average of an array
6. Input/Output String
7. Solving mathematical problems using user defined functions

Post Lab Exercise and Further Readings:
Through review of C/C++ for coding

Session 2: Introduction to String Operations and Tokenization

Intended Learning Outcome:
a. Solving programming problems with Array, Function, Recursion, Pointer

Expected skills:
a. Solving various simple computing problems

Tools Required:
a. CodeBlocks

Session Detail:

1. Input two string from user
2. Determine the length of each string and display
3. Concatenate the strings into one
4. Determine the final length of the string
5. Split the words with meaningful token from the string

Post Lab Exercise and Further Readings:
Through review of C/C++ for coding

Session 3: Tokenization

Intended Learning Outcome:
a. Understand the basics of sorting algorithms
b. Students learn to tokenize from a string, which describes how to break the program into pieces to work with compiler. In lexical analysis, tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining.

Expected skills:
a. Students will be able to implement how to tokenize multiple string using C/C++ .

Tools Required:
a. CodeBlocks

Session Detail:
a) Input strings from user.
b) Split the words depending on the delimiters.
c) Do the same task using strtok() built-in functions.

Sample Input:	**Sample Output:**
Hello DIU Students	Hello DIU Students

Post Lab Exercise:
Tokenize various types' code.

Further Readings:
White Space Detection, Counting and Removal

Session 4: White Space Detection, Counting and Removal

Intended Learning Outcome:
a. Understand the basics of lexical analyzer.
b. Removing White Spaces are the secondary role of lexical analyzer.
c. Students learn to detect the different types White Spaces (Spaces, Tabs, newlines), count number of White Spaces and remove all the white spaces.

Expected skills:
a. Students will be able to implement how to detect, count and remove white spaces from code using C/C++ .

Tools Required:
a. CodeBlocks

Session Detail:
a) Input strings from file having White Spaces.
b) Detect white spaces, Count number of white spaces and remove the white spaces

Sample Input:	**Sample Output:**

int x, y;
x = 4;
y = 5;
printf("x=%d",&x);
printf("y=%d",&y);
return 0;
	int x, y;
	x = 4;
	y = 5;
	printf("x=%d",&x);
	printf("y=%d",&y);
	return 0;

Post Lab Exercise:
Tokenize various types' code.

Further Readings:
Comment Detection and Removal

Session 5: Comment Detection and Removal

Intended Learning Outcome:
a. Understand the basics of lexical analyzer.
b. Detecting Comment and removing it is the secondary role of lexical analyzer.
c. Students learn to detect the different types comments (Single line and Multiline Comment).

Expected skills:
a. Students will be able to implement how to detect, count and remove comments from code using C/C++ .

Tools Required:
a. CodeBlocks

Session Detail:
a) Input strings from file having comments.
b) Detect single line and multiline comments, count number of comments and remove the comments.

Sample Input:	**Sample Output:**

int x, y; //declaring variables
x = 4; //initializing variables
y = 5; //initializing variables
/*display the values*/
printf("x=%d",&x);
printf("y=%d",&y);
return 0;
	int x, y;
	x = 4;
	y = 5;
	printf("x=%d",&x);
	printf("y=%d",&y);
	return 0;

Post Lab Exercise:
Tokenize various types' code.

Further Readings:
Symbol table

Session 6: Symbol Table Generation

Intended Learning Outcome:

a. Understand the basics of Symbol table
b. Symbol table is collection of different types of tokens of a program. Symbol table is an important data structure created and maintained by compilers in order to store information about the occurrence of various entities such as variable names, function names, objects, classes, interfaces, etc. Symbol table is used by both the analysis and the synthesis parts of a compiler.

Expected skills:

a. Students will be able to generate symbol table using C/C++

Tools Required:

a. CodeBlocks

Session Detail:

Sample Run of previous works that is either supporting Symbol table management

Post Lab Exercise:
Implement symbol table for various program

Further Readings:
Syntax analyzing

Session 7: Designing Lexical Analyzer

Intended Learning Outcome:
a. Designing the first phase of the Compiler.
b. Understanding the primary and secondary roles of Lexical Analyzer.
c. Incorporate the tokenization, comment detection and removal, white space detection and removal into one.

Expected skills:
a. Students will be able to solve various problems using the knowledge of previous labs and implement the solution using C/C++ code

Tools Required:
a. CodeBlocks

Session Detail:
a) Generate an appropriate code of visualize the implementation of the mentioned task.

Post Lab Exercise:
Recognize strings for different CFG grammar

Further Readings:
Regular Expression

Session 8: Regular Expression

Intended Learning Outcome:
a. Understand how regular expression works
b. Use of regular expression for constructing a recognizer.
c. Understanding the validity and invalidity of a recognizer.

Expected skills:
a. Students will be able to solve various problems using the knowledge of regular expression and implement the solution using C/C++ /python code.

Tools Required:
a. CodeBlocks

Session Detail:
a) Generate a recognizer that can recognize the input strings of "a*abb" expression.
b) Students may use the "RegEx" package of python for doing so.

Post Lab Exercise:
Recognize strings for different regular expression

Further Readings:
First and Follow Function

Session 9: Calculate FIRST

Intended Learning Outcome:
a. Understand the basics of FIRST calculation from Context Free Grammar
b. The construction of both top-down and bottom-up parsers is aided by two functions, FIRST and FOLLOW, associated with a grammar. For a preview of how FIRST can be used during predictive parsing, consider two productions of A where, $A \rightarrow \alpha \mid \beta$, where FIRST($\alpha$) and FIRST($\beta$) are disjoint sets. Student learns the parsing.

Expected skills:
a. Students will be able to generate FIRST using C/C++ code

Tools Required:
a. CodeBlocks

Session Detail:
a) Calculate FIRST from theoretical understanding by following First and Follow rules.
b) Verifying the program works correct

Post Lab Exercise: Take home assignment and online submission of solution to google class room.

Further Readings: Can you implement other algorithms.

Session 10: Calculate Follow

Intended Learning Outcome:
a. Understand the basics of FOLLOW generation
b. FOLLOW associated with formal grammar in compiler design. It's a type of parsing. The construction of both top-down and bottom-up parsers is aided by two functions, FIRST and FOLLOW

Expected skills:
a. Students will be able implement Follow of a context free grammar using C/C++ code

Tools Required:
a. CodeBlocks

Session Detail:
a) Calculate FOLLOW from theoretical understanding by following First and Follow rules.
b) Verifying they are working

Post Lab Exercise: Take home assignment and online submission of solution to google class room.

Further Readings: Can you implement other algorithms.