Compiler Construction (327106) Assignment No. 2

Due Date/Time	30th Dec, 2022 11:59 PM
Files to be submitted	Documentation of each module along with code file.
File Naming	Roll_No_Assign_02.pdf Roll_No_Assign_02.py Note: Any assignment that is not according to format will be marked as zero.
Coding Guides	1. Use of proper variable declaration/initialization according to the naming conventions (**camelCase, snake_case, PascalCase**) 2. Use of proper function for each module. Note: Marks will be deducted if not following the above guide line.
Submission Guide line	**1.** Code along with documentations should be submitted on teams by due date/time. **2.** Also update your solution on your **Github** profile. **3.** Also include your **Github** link in the documentation.
Plagiarism	Any kind of plagiarism will result in F grade in course

Compiler operates in various phases each phase transforms the source program from one representation to another. Every phase takes inputs from its previous stage and feeds its output to the next phase of the compiler.

There are 6 phases in a compiler. Each of this phase help in converting the high-level langue the machine code. The phases of a compiler are:

1. Lexical analysis
2. Syntax analysis
3. Semantic analysis
4. Intermediate code generator
5. Code optimizer
6. Code Generator

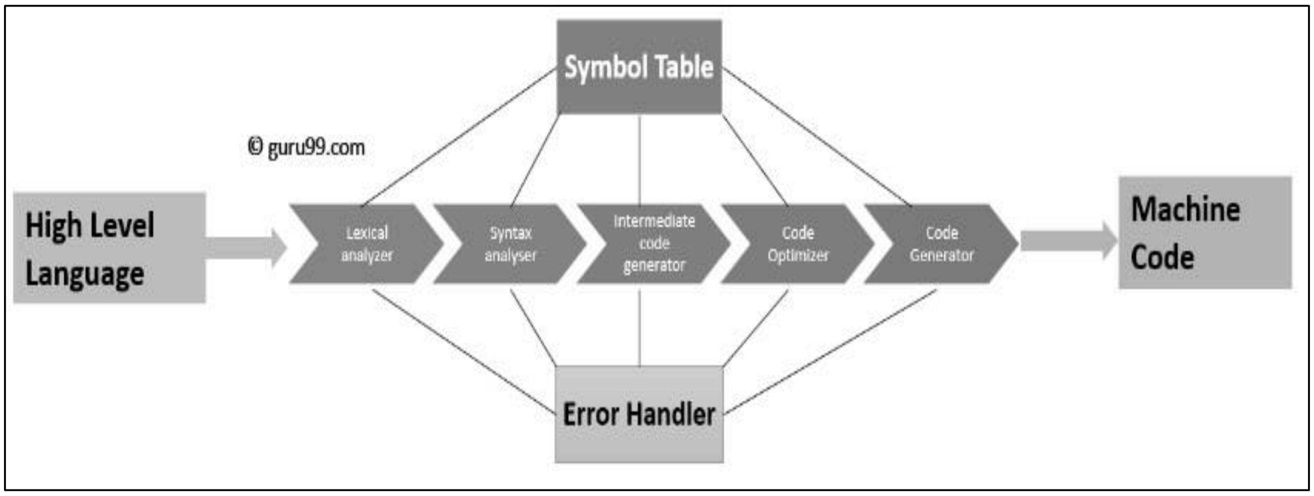

Figure 1 Phases of Compiler

All these phases convert the source code by dividing into tokens, creating parse trees, and optimizing the source code by different phases. You are required to implement all the phases with module wise. For this assignment you are going to implement two phases:

Module 1:

Implementation of lexical analyzer

- Tokenization of expression (expression can be i.e a + (b*c) or 3+ (5*2) digits, alphabets, characters)
- Building regex for the expression
- Output tags/ tokens of the expression (i.e. ['a', '+', '(', 'b', '*', 'c', ')']) **Note**: For this task you are required to explore python re library:

  1. https://docs.python.org/3/library/re.html
  2. https://www.w3schools.com/python/python_regex.asp

Module 2:

Implementation of syntax tree using AST library of python

Note: For this task you are required to explore python AST library:

1. https://docs.python.org/3/library/ast.html
2. https://www.pythonpool.com/python-ast/

Code:

```python
import re
import ast
from ast import parse

def tokenize(expression):
  # Regular expression for all types of tokens
  pattern = r'[0-
9_]|[abcdefghijklmnopqrstuvwxyz_]\d*|[ABCDEFGHIJKLMNOPQRSTUVWXYZ_]\d*|[+*-
/]|[~`!@#$%^&()_=|";:<>,.?]'

  # Tokenizing the input with regular expression.
  tokens = re.findall(pattern, expression)

  return tokens

def parse_tree(expression):
  tree = ast.parse(expression, mode='exec')
  print(ast.dump(tree))

# Tests the tokenize function
expression = "Ali@li-reg+27"
print(tokenize(expression))
print(parse_tree(expression))
```

END