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', ')'])

Module 2:

Implementation of syntax tree using AST library of python

Code:

```
import  re
string = [ " ( a + ( b * c ) + a ^ * ) " ]
print("Input string : " + str(string))
re_string = [sub.split() for sub in string]

import ast
print(" Tokenized string : " + str(re_string))
code = ast.parse( "print ( a + ( b * c ) + a )" )
print(ast.dump(code))
```