CIIC 4030/ICOM 3046 Programming Languages
Assignment #3

This assignment will expand the PLY-based parser in assignment #2 to structured parsing, where the parser builds an **abstract syntax tree (AST),** an intermediate representation. For each rule, write the corresponding intermedia representation. There are at least three forms of representation:

1. **Direct Evaluation**. For example,

```
def p_stm_enclosed(p):
    'stm : LPAREN stm RPAREN'
    p[0] = p[2]
```

   This means the parentheses are ignored in the final syntax tree. The enclosed statement (p[2]) is directly returned as the result.

2. **Further Evaluation**. For example, in the following rule

```
def p_stm_let(p):
    'stm : LET facts IN stm END'
    p[0] = {'type': 'stm_let', 'facts': p[2], 'stm': p[4]}
```

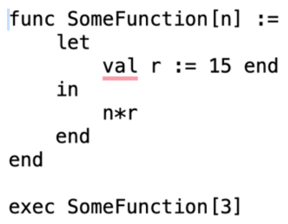
The semantics meaning is that it adds the following items to the abstract syntax tree: 'type': 'stm_let', which identifies a let statement; 'facts': p[2], which captures the bindings (declarations) before IN; and 'stm': p[4], which captures the body (the statement inside IN ... END).

3. **Recursive Evaluation**. For example,

```
def p_args1(p):
    '''args : ID_FUNC COMMA args'''
    p[0] = [{'type': 'id_func', 'id_func': p[1]}] + p[3]
```

{'type': 'id_func', 'id_func': p[1]}, wraps the function identifier in a dictionary. + p[3], recursively appends the following parsed argument list. This rule allows parsing a comma-separated list of function identifiers. It recursively constructs a list, ensuring each ID_FUNC is stored in a structured way. It uses a base case rule (e.g., args : ID_FUNC) to terminate recursion. The result is a list of dictionaries, each representing an argument.

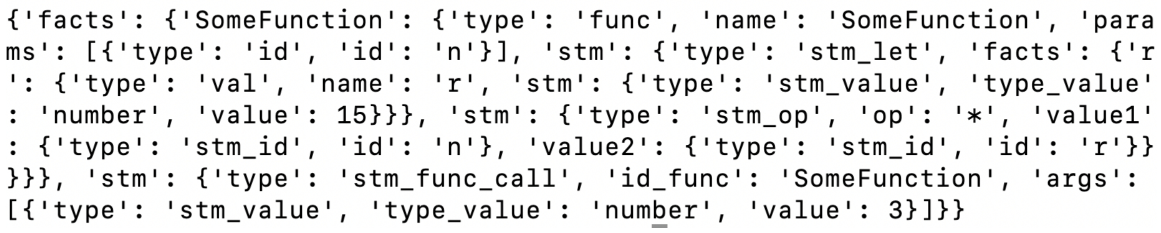
If your input is the following code



```
func SomeFunction[n] :=
    let
        val r := 15 end
    in
        n*r
    end
end

exec SomeFunction[3]
```

The AST produced by the parser must be something like



```
{'facts': {'SomeFunction': {'type': 'func', 'name': 'SomeFunction', 'para
ms': [{'type': 'id', 'id': 'n'}], 'stm': {'type': 'stm_let', 'facts': {'r
': {'type': 'val', 'name': 'r', 'stm': {'type': 'stm_value', 'type_value'
: 'number', 'value': 15}}}, 'stm': {'type': 'stm_op', 'op': '*', 'value1'
: {'type': 'stm_id', 'id': 'n'}, 'value2': {'type': 'stm_id', 'id': 'r'}}
}}}, 'stm': {'type': 'stm_func_call', 'id_func': 'SomeFunction', 'args':
[{'type': 'stm_value', 'type_value': 'number', 'value': 3}]}}
```

You must submit all the files required to run the parser in Moodle.