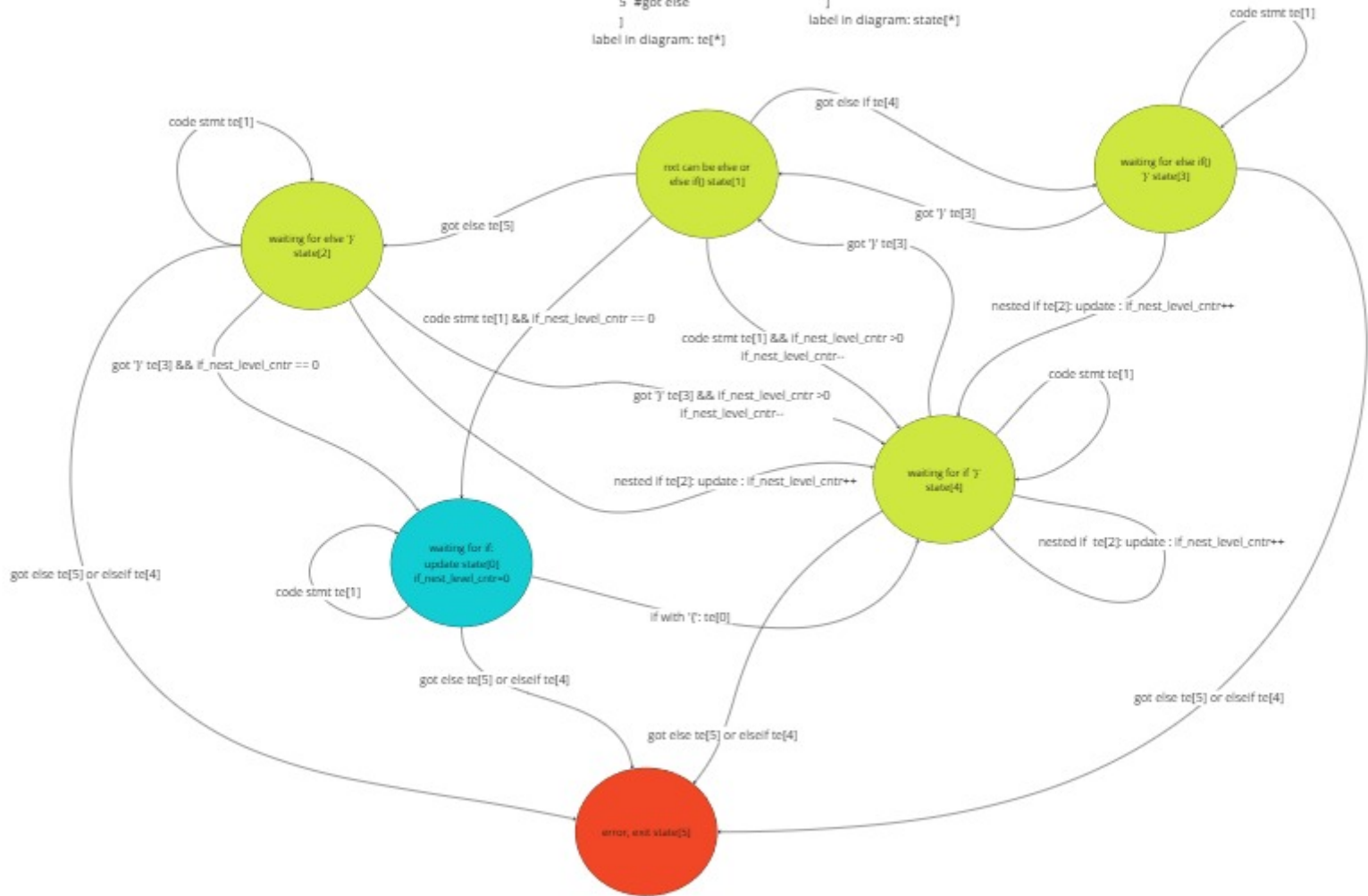


```
fsm_transition_events = [
0, #'if with '
1, #'any code stmt
2, #'nested if
3, #'got ')'
4, #'got else if
5, #'got else
]
label in diagram: te[*]
```

```
fsm_states = [0, #'waiting for ' if ',
1, #'next can be else or else if',
2, #'waiting for else cls brace',
3, #'waiting for else if cls brace',
4, #'waiting for if cls brace',
5, #'error, exit'
]
label in diagram: state[*]
```







pedmas_assembler() called

expn = "a+b+c*d-f*g/h+w%d^x/y*2^1-z*q^3%e"

loop through each operator in precedence order (prob_obj attribute:
operator_lst_precedence_h2l)

operator_lst_precedence_h2l = ['^', '/', '*', '%', '+', '-']

chk if operation is '%': reqd. for purpose of proper functioning of regex

yes

no

find sections of expn with current operation: **do not** use escape sequence char in regex checking str

find sections of expn with current operation: use escape sequence char in regex checking str

break up each sub expn into tuples of vars creating a list of tuples

for op == '%': [(2, 1), (q, 3)]

loop through the var tuple list

put tuple contents in a list

loop through the tuple's var list

pop tuple from front twice to get two operands

(a, b, c) => var1 = a, var2 = b, tuple = (c)

create temp var name and then create replacement string from the popped vars

push the newly created temp var name into front of tuple

update parser sequence event attribute of the prob_obj

replace the sub expn str with temp var name in the main expn str

update tmp_var_cntr attribute of prg_obj

return expn to calling func.: expn_seq_assembler()

all tuples in tuple not yet processed

all vars in tuple not yet processed

vars not yet processed

