Compiler Construction
Professor Jeremy Siek

Quiz 2
September 19, 2012

Name: _____

This exam has 3 questions, for a total of 10 points.

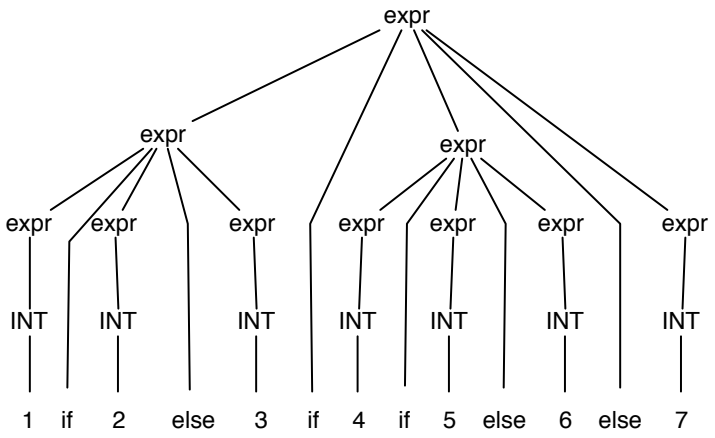
1. [3 points] Given the following grammar for a subset of Python:

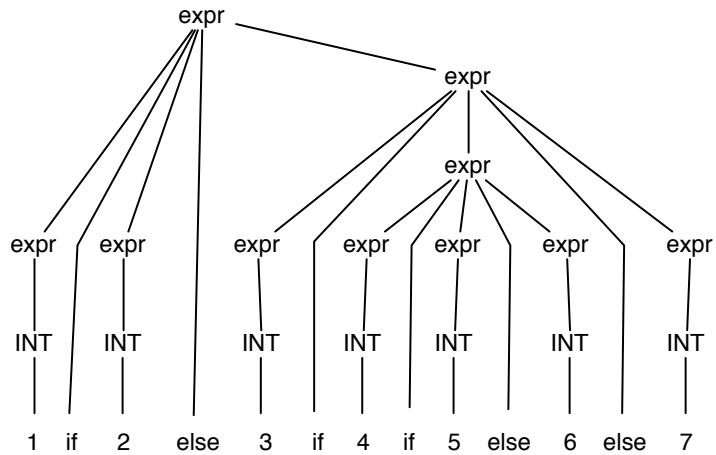
```
(1)  expr ::= INT
(2)  expr ::= expr IF expr ELSE expr
(3)  expr ::= expr PLUS expr
```

draw all of the parse trees for the input:

```
1 if 2 else 3 if 4 if 5 else 6 else 7
```

Solution:





2. [3 points] Write the PLY parser specification for the grammar in question 1. Skip the lexer specification and assume the `INT`,`PLUS`,`IF`, and `ELSE` tokens have already been defined. Write the `p_` functions and the `precedence` variable (using Python's precedence rules). The actions inside the `p_` functions should build the appropriate Python AST nodes. Hint: the AST class for the if-else expression is shown below.

```
class IfExp(Node):
    def __init__(self, test, then, else_):
        ...
```

Solution:

```
def p_int_expr(p):
    'expr : INT'
    p[0] = Const(p[1])

def p_if_expr(p):
    'expr : expr IF expr ELSE expr
    p[0] = IfExp(p[3], p[1], p[5])
```

```
def p_plus_expr(p):
    'expr : expr PLUS expr'
    p[0] = Add((p[1], p[3]))

precedence = (('nonassoc', 'IF', 'ELSE'),
              ('left', 'PLUS'), )
```

3. [4 points] Given the below grammar and parse table, list the states, stack configurations, and actions (shift,reduce,goto) that occur while parsing the input "`aabb`".

```
state 0
    start ::= . match
    match ::= . "a" "b"
    match ::= . "a" match "b"
    on "a" shift to state 1
    on match goto state 2
state 1
    match ::= "a" . "b"
    match ::= "a" . match "b"
    match ::= . "a" "b"
    match ::= . "a" match "b"
    on "a" shift to state 1
    on "b" shift to state 3
    on match goto state 4
```

```
(1) start ::= match
(2) match ::= "a" "b"
(3) match ::= "a" match "b"

state 2, accept
    start ::= match .
state 3
    match ::= "a" "b" .
    on end,"a", or "b" reduce by rule 2
state 4
    match ::= "a" match . "b"
    on "b" shift to state 5
state 5
    match ::= "a" match "b" .
    on end,"a", or "b" reduce by rule 3
```

Solution:

State	Stack	Input Left	Action
0	[]	aabb	Shift to state 1
1	[(1,a)]	abb	Shift to state 1
1	[(1,a),(1,a)]	bb	Shift to state 3
3	[(3,b),(1,a),(1,a)]	b	Reduce by rule 2 to state 1, goto 4
4	[(4,match),(1,a)]	b	Shift to state 5
5	[(5,b),(4,match),(1,a)]		Reduce by rule 3 to state 0, goto 2
2	[(2,match)]		