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.

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

**Solution:**

```python
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])

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”.

```plaintext|
state 0
state 1
state 2, accept
state 3
state 4
state 5
|
start ::= . match
match ::= . "a" "b"
match ::= . "a" match "b"
on "a" shift to state 1
on match goto state 2
match ::= "a" . "b"
match ::= "a" . match "b"
match ::= . "a" "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
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:**

<table>
<thead>
<tr>
<th>State</th>
<th>Stack</th>
<th>Input Left</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[]</td>
<td>aabb</td>
<td>Shift to state 1</td>
</tr>
<tr>
<td>1</td>
<td>[(1,a)]</td>
<td>abb</td>
<td>Shift to state 1</td>
</tr>
<tr>
<td>1</td>
<td>[(1,a),(1,a)]</td>
<td>bb</td>
<td>Shift to state 3</td>
</tr>
<tr>
<td>3</td>
<td>[(3,b),(1,a),(1,a)]</td>
<td>b</td>
<td>Reduce by rule 2 to state 1, goto 4</td>
</tr>
<tr>
<td>4</td>
<td>[(4,match),(1,a)]</td>
<td>b</td>
<td>Shift to state 5</td>
</tr>
<tr>
<td>5</td>
<td>[(5,b),(4,match),(1,a)]</td>
<td>b</td>
<td>Reduce by rule 3 to state 0, goto 2</td>
</tr>
<tr>
<td>2</td>
<td>[(2,match)]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>