Answers:

1(a). $P(A) \equiv \exists p \forall s \in A, |s| \geq p \longrightarrow (\exists u, v, x, y, z$ such that $s = uvxyz$, $|vxy| \leq p$, $vy \neq \epsilon$ and $\forall k \geq 0$, $uv^k xy^k z \in A)$.

1(b). We want to show that $P(L)$ is true. Let $p = 5$. Suppose s is in L and has at least 5 symbols. Thus, s is of the form $a^i b^i c^j$.

Case 1. Suppose $i - 1 > j$. Let $u = a^{i-1}, v = a, x = \epsilon, y = b, z = b^{i-1}c^j$.

Case 2. Suppose $i - 1 = j$. Since $|s| \geq 5$, we deduce that $i \geq 2$. Let $u = a^{i-2}, v = aa, x = \epsilon, y = bb, z = b^{i-2}c^j$.

Case 3. Suppose $i < j - 1$. Let $u = a^i b^i c^{j-1}, v = c, x = \epsilon, y = \epsilon, z = \epsilon$.

Case 4. Suppose $i = j - 1$. Let $u = a^i b^i c^{j-2}, v = cc, x = \epsilon, y = \epsilon, z = \epsilon$.

In the above cases, one can verify that for all $k \geq 0, uv^k xy^k z \in L$.

2(a).

We can have two parse trees for the sentence `if y==1 then if y==1 then x=1 else x=1`. Following are two rightmost derivations for the sentence.

```
<stmt> => <if-then>
       => if <condition> then <stmt>
       => if y==1 then <stmt>
       => if y==1 then <if-then-else>
       => if y==1 then if <condition> then <stmt> else <stmt>
       => if y==1 then if y==1 then <stmt> else <stmt>
       => if y==1 then if y==1 then x=1 else <stmt>
       => if y==1 then if y==1 then x=1 else x=1

<stmt> => <if-then-else>
       => if <condition> then <stmt> else <stmt>
       => if y==1 then <stmt> else <stmt>
       => if y==1 then <if-then> else <stmt>
       => if y==1 then if <condition> then <stmt> else <stmt>
       => if y==1 then if y==1 then <stmt> else <stmt>
       => if y==1 then if y==1 then x=1 else <stmt>
       => if y==1 then if y==1 then x=1 else x=1
```

2(b)

Let us denote `if y==1 then` by `'i'`, `x=1` by `'a'` and `else` by `'e'`.

The PDA design is summarized in the following C-style codes: (It is better to explain the ideas in plain English then giving the codes)

```
while ((c=getChar()) != EOF)
   if (c=='i')
        if (top()=='is') reject; // syntax error
        else push 'i'
   else if (c=='a')
        if (top()=='is') reject; // syntax error
        else {
               while (top() == 'ise') pop 'ise';  // s ->if-cond-then s else s
               if (isEmptyStack())
                  if (getChar()==EOF) accept; // assume input is correct
                  else reject; // syntax error
               else if (top()=='i') {pop 'i'; push 'is'}
             }
   else if (top()=='is') {pop 'is'; push 'ise';}
        else reject; // syntax error
if (top() != 'is') reject; // syntax error
pop 'is';
while (!isEmptyStack())
   if (top()=='i') pop 'i';        // s -> if-cond-then s
   else if (top()=='ise') pop 'ise'; // s -> if-cond-then s else s
   else if (top()=='is') reject; // syntax error
```

3(a). The transitions are
$(q_0, 0, q0), (q0, 1, q1)$
$(q_1, 0, q0), (q1, 1, q2)$
$(q_2, 0, q3), (q2, 1, q2)$
$(q_3, 0, q0), (q3, 1, q4)$
$(q_4, 0, q0), (q4, 1, q5)$
$(q_5, 0, q3), (q5, 1, q6)$
$(q_6, 0, q6), (q6, 1, q6)$

3(b). Same as the automaton for 3(a) except that the transitions from $q6$ become $(q_6, 0, q3), (q6, 1, q2)$.