1. (a)

Here is the AST, with one node on each numbered line (arbitrarily numbered breadth-first).

```
1: let f
   / \ 
 2: fun x \
3: | fun y
4: x |
5: fun z
   | 
6: if
   / | \
7: y | \
8: z \ 
9: @
   / \
10: f \ 
11: 3
```

From this tree, we generate the following constraints:

<table>
<thead>
<tr>
<th>Node #</th>
<th>Rule</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>let</td>
<td>t2 = tf and t1 = t3</td>
</tr>
<tr>
<td>2</td>
<td>fun</td>
<td>t2 = tx -&gt; t4</td>
</tr>
<tr>
<td>3</td>
<td>fun</td>
<td>t3 = ty -&gt; t5</td>
</tr>
<tr>
<td>4</td>
<td>var</td>
<td>t4 = tx</td>
</tr>
<tr>
<td>5</td>
<td>fun</td>
<td>t5 = tz -&gt; t6</td>
</tr>
<tr>
<td>6</td>
<td>if</td>
<td>t7 = Bool and t6 = t8 = t9</td>
</tr>
<tr>
<td>7</td>
<td>var</td>
<td>t7 = ty</td>
</tr>
<tr>
<td>8</td>
<td>var</td>
<td>t8 = tz</td>
</tr>
<tr>
<td>9</td>
<td>app</td>
<td>t10 = t11 -&gt; t9</td>
</tr>
<tr>
<td>10</td>
<td>var</td>
<td>t10 = tf</td>
</tr>
<tr>
<td>11</td>
<td>int</td>
<td>t11 = Int</td>
</tr>
</tbody>
</table>

We can solve this by inspection:

First, using the identities for t2, t4, t7, t8, t10, t11 we can substitute for these variables, leading to the following modified constraints:

```
2'     tf = tx -> tx
6'     ty = Bool and t6 = tz = t9
9'     tf = Int -> t9
```

Using t1 = t3, we can substitute for t3 to get the modified constraint

```
3'     t1 = ty -> t5
```
(Choosing whether to get rid of \( t1 \) or \( t3 \) is fairly arbitrary, but we ultimately want to know the root expression type \( t1 \), so we keep that.)

Similarly, from \( t6 = tz = t9 \), we can substitute for \( t6 \) and \( t9 \) (again fairly arbitrary, but we ultimately want to know \( tz \)), getting

\[
5' \quad t5 = tz \to tz \\
9'' \quad tf = Int \to tz
\]

Equating the two expressions (2' and 9'') for \( tf \), we get that

\[
tx \to tx = Int \to tz
\]

which implies that \( tx = tz = Int \).

Summarizing, we have

\[
t1 = \text{Bool} \to (\text{Int} \to \text{Int}) \\
tf = \text{Int} \to \text{Int} \\
tx = \text{Int} \\
ty = \text{Bool} \\
tz = \text{Int}
\]

(b)

The AST:

```
1: fun f
   | 2: fun g
   | 3: fun x
   | 4: @
      / \ 5: f \ 6: @
      / \ 7: g \ 8: x
```

The generated constraints:

<table>
<thead>
<tr>
<th>Node #</th>
<th>Rule</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fun</td>
<td>( t1 = tf \to t2 )</td>
</tr>
<tr>
<td>2</td>
<td>fun</td>
<td>( t2 = tg \to t3 )</td>
</tr>
<tr>
<td>3</td>
<td>fun</td>
<td>( t3 = tx \to t4 )</td>
</tr>
<tr>
<td>4</td>
<td>app</td>
<td>( t5 = t6 \to t4 )</td>
</tr>
</tbody>
</table>
5 \text{ var } t5 = tf
6 \text{ app } t7 = t8 \rightarrow t6
7 \text{ var } t7 = tg
8 \text{ var } t8 = tx

Solution by inspection:

Using the identities from nodes 5, 6, and 7, we can rewrite the contraints for nodes 4 and 6 as

4’ \quad tf = t6 \rightarrow t4
6’ \quad tg = tx \rightarrow t6

There are no other contraints on \( tx, t4 \) and \( t6 \), so the overall type must be parametric in (i.e. polymorphic over) these types. The type of the overall expression is

\[
\text{t1 = tf} \rightarrow \text{t2} \quad \text{ (by 1)}
= \text{tf} \rightarrow (\text{tg} \rightarrow \text{t3}) \quad \text{ (by 2)}
= \text{tf} \rightarrow (\text{tg} \rightarrow (tx \rightarrow t4)) \quad \text{ (by 3)}
= (t6 \rightarrow t4) \rightarrow ((tx \rightarrow t6) \rightarrow (tx \rightarrow t4)) \quad \text{ (by 4’ and 6’)}
\]

Or, using more suggesting names for the polymorphic types, and the convention that \( \rightarrow \) associates to to the right:

\[
\text{t1 = (tb} \rightarrow \text{tc}) \rightarrow (\text{ta} \rightarrow \text{tb}) \rightarrow (\text{ta} \rightarrow \text{tc})
tx = \text{ta}
tf = \text{tb} \rightarrow \text{tc}
tg = \text{ta} \rightarrow \text{tb}
\]

which makes sense for a general-purpose “compose” function.