For the following expression:

\[ g(x, y) = (x' + y)' \]

Draw a circuit for \( g(x, y) \):

Fill out a truth table for \( g(x, y) \). (We’ve provided extra columns to compute \( x' \) and \( x' + y \).)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>g(x,y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
### Verilog Modules

```verilog
module sc_block(s, c, a, b);
    output s, c;
    input a, b;
    wire w1, w2, not_a, not_b;

    // the "c" output is just the AND of the two inputs
    and a1(c, a, b);

    // the "s" output is 1 only when exactly one of the inputs is 1
    not n1(not_a, a);
    not n2(not_b, b);
    and a2(w1, a, not_b);
    and a3(w2, b, not_a);
    or o1(s, w1, w2);

endmodule // sc_block
```

Draw a circuit diagram for the Verilog module above:

![Circuit Diagram](image)

**Figure 1.** AND, OR, and NOT gates.
Hierarchical design: building modules from modules

![Diagram of sc2_block implementation using sc_block](image)

**Figure 2.** The sc2_block, which is implemented using sc_block, and its partially completed truth table.

Complete the truth table and write the Verilog for the `sc2_block` module: