1) Review the following C code segments. For each segment, discuss and write down “what is the state of these programs?” How/when is the state changing?

a)

```c
int variable1 = 3;
int variable2 = 8;

variable2 = variable2 + variable1;

int variable3 = variable2 + 5;
```

b)

```c
if (x >= 0) {
    x = x + 5;
}
x |= 1;
```

void init_array_of_length_12(int array[12]) {
    for (int i = 0 ; i <= 12 ; i++) {
        array[i] = 0;
    }
}
For the following expression:

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

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

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

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

![Diagram of sc2_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: