```
==> vending_machine.v <==
module d_flip_flop (
    input clk,
    input rst,
    input data,
    output reg q
);
  always @(posedge clk or rst) begin
    #1 q = (rst) ? ~rst : data;
  end
endmodule

module vending_machine (
    input coin_type,
    input clk,
    input rst,
    output is_dispensed,
    output [1:0] state
);
  d_flip_flop U0 (
      .clk(clk),
      .rst(rst),
      .data(state[1] | coin_type & state[0]),
      .q(is_dispensed)
  );
  d_flip_flop U1 (
      .clk(clk),
      .rst(rst),
      .data(~coin_type & ~state[1]),
      .q(state[0])
  );
  d_flip_flop U2 (
      .clk(clk),
      .rst(rst),
      .data((~coin_type & ~state[1] & state[0]) + (coin_type & ~state[0])),
      .q(state[1])
  );
endmodule

==> tb.py <==
import random

code_template = (
    lambda start, test_cases: f"""
module orange_tb();
    initial begin
        $dumpfile("orange_tb.vcd");
        $dumpvars(0, orange_tb);
    end

    reg coin, clk, rst;
    wire [1:0] state;
    wire dispenser_output;

    initial begin
        clk = 1'b0;
        rst = 1'b1;
        coin = 1'bz;
        $display("----RESET----");
{start}
        #10 rst = 1'b0;
        $display("----RESET----");
{test_cases}
        #11 $finish(0);
    end

    always forever begin
        #5 clk = ~clk;
        if (clk) begin
            $display("inp: %b :: out: %b :: state: %2b :: rst: %b", coin, dispenser_output, state, rst);
        end
    end

    vending_machine U0 (
        .coin_type(coin),
        .clk(clk),
```

```
        .rst(rst),
        .is_dispensed(dispenser_output),
        .state(state)
    );
endmodule
"""
)

# Coin Sequence Length
N = 5
valid_values = [[0, 1], [1, 0], [0, 0, 0]]
numbers = []
for _ in range(N):
    numbers.extend(random.choice(valid_values))
print(numbers)

tests = [f"\t#10 coin=1'b{x};" for x in numbers]

with open("orange_tb.v", "w") as testbench:
    print(code_template(tests[0], "\n".join(tests[1:])), file=testbench)

==> orange_tb.v <==

module orange_tb();
    initial begin
        $dumpfile("orange_tb.vcd");
        $dumpvars(0, orange_tb);
    end

    reg coin, clk, rst;
    wire [1:0] state;
    wire dispenser_output;

    initial begin
        clk = 1'b0;
        rst = 1'b1;
        coin = 1'bz;
        $display("----RESET----");
 #10 coin=1'b0;
        #10 rst = 1'b0;
        $display("----RESET----");
 #10 coin=1'b1;
 #10 coin=1'b0;
 #10 coin=1'b1;
 #10 coin=1'b0;
 #10 coin=1'b1;
 #10 coin=1'b0;
 #10 coin=1'b0;
 #10 coin=1'b0;
 #10 coin=1'b0;
 #10 coin=1'b0;
 #10 coin=1'b0;
        #10 $finish(0);
    end

    always forever begin
        #5 clk = ~clk;
        if (clk) begin
            $display("inp: %b :: out: %b :: state: %2b :: rst: %b", coin, dispenser_output, state, rst);
        end
    end

    vending_machine U0 (
        .coin_type(coin),
        .clk(clk),
        .rst(rst),
        .is_dispensed(dispenser_output),
        .state(state)
    );
endmodule


==> output.txt <==
VCD info: dumpfile orange_tb.vcd opened for output.
----RESET----
inp: z :: out: 0 :: state: 00 :: rst: 1
```

```
inp: 0 :: out: 0 :: state: 00 :: rst: 1
----RESET----
inp: 0 :: out: 0 :: state: 01 :: rst: 0
inp: 1 :: out: 0 :: state: 11 :: rst: 0
inp: 0 :: out: 1 :: state: 00 :: rst: 0
inp: 1 :: out: 0 :: state: 01 :: rst: 0
inp: 0 :: out: 1 :: state: 00 :: rst: 0
inp: 1 :: out: 0 :: state: 01 :: rst: 0
inp: 0 :: out: 1 :: state: 00 :: rst: 0
inp: 0 :: out: 0 :: state: 01 :: rst: 0
inp: 0 :: out: 0 :: state: 11 :: rst: 0
inp: 0 :: out: 1 :: state: 00 :: rst: 0
inp: 0 :: out: 0 :: state: 01 :: rst: 0
inp: 0 :: out: 0 :: state: 11 :: rst: 0
```