```vhdl
----------------------------------------------------------------------------------
-- Name: Capt Jeff Falkinburg
-- Date: Spring 2016
-- Course: ECE 281
-- File: Lsn20_stoplight_tb.vhd
-- HW:   Lecture 20
-- Purp: Testbench for Basic Stoplight State Machine
-- Doc:  None
-- Academic Integrity Statement: I certify that, while others may have
-- assisted me in brain storming, debugging and validating this program,
-- the program itself is my own work. I understand that submitting code
-- which is the work of other individuals is a violation of the honor
-- code.  I also understand that if I knowingly give my original work to
-- another individual is also a violation of the honor code.
----------------------------------------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY stoplight_tb IS
END stoplight_tb;

ARCHITECTURE behavior OF stoplight_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT stoplight
    PORT(
         C : IN  std_logic;
         reset : IN  std_logic;
         clk : IN  std_logic;
         R : OUT  std_logic;
         Y : OUT  std_logic;
         G : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal C : std_logic := '0';
   signal reset : std_logic := '0';
   signal clk : std_logic := '0';

   --Outputs
   signal R : std_logic;
   signal Y : std_logic;
   signal G : std_logic;

   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN

```

```
   -- Instantiate the Unit Under Test (UUT)
   uut: stoplight PORT MAP (
          C => C,
          reset => reset,
          clk => clk,
          R => R,
          Y => Y,
          G => G
        );

   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 20 ns;

      reset <= '1';
      wait for 20 ns;

      reset <= '0';
      wait for 20 ns;

      C <= '1';
      wait for 40 ns;

      C <= '0';
      wait for 40 ns;

      wait for clk_period*10;

      -- insert stimulus here

      wait;
   end process;

END;

```