```
library IEEE;
use IEEE.std_logic_1164.all;
library UTILS;
use UTILS.io1164.all;
use std.textio.all;

entity loadCntTB is
end loadCntTB;

architecture testbench of loadCntTB is

  component loadCnt port (
    data: in std_logic_vector (7 downto 0);
    load: in std_logic;
    clk: in std_logic;
    rst: in std_logic;
    q: out std_logic_vector (7 downto 0)
    );
  end component;

file vectorFile: text is in "vectorfile";
type vectorType is record
  data: std_logic_vector(7 downto 0);
  load: std_logic;
  rst: std_logic;
  q: std_logic_vector(7 downto 0);
end record;

signal testVector: vectorType;
signal TestClk: std_logic := '0';
signal Qout: std_logic_vector(7 downto 0);

constant ClkPeriod: time := 100 ns;

for all: loadCnt use entity work.loadcnt(rtl);

begin

-- File reading and stimulus application
  readVec: process
    variable VectorLine: line;
    variable VectorValid: boolean;
    variable vRst: std_logic;
    variable vLoad: std_logic;
    variable vData: std_logic_vector(7 downto 0);
    variable vQ: std_logic_vector(7 downto 0);

  begin
    while not endfile (vectorFile) loop
      readline(vectorFile, VectorLine);

      read(VectorLine, vRst, good => VectorValid);
      next when not VectorValid;
      read(VectorLine, vLoad);
      read(VectorLine, vData);
      read(VectorLine, vQ);

      wait for ClkPeriod/4;

      testVector.Rst <= vRst;
      testVector.Load <= vLoad;
      testVector.Data <= vData;
      testVector.Q <= vQ;

      wait for (ClkPeriod/4) * 3;
```

```
    end loop;

  assert false
    report "Simulation complete"
    severity note;

  wait;

  end process;

-- Free running test clock
  TestClk <= not TestClk after ClkPeriod/2;

-- Instance of design being tested
  u1: loadCnt port map (Data => testVector.Data,
                        load => testVector.Load,
                        clk => TestClk,
                        rst => testVector.Rst,
                        q => Qout
                       );

-- Process to verify outputs
  verify: process (TestClk)
  variable ErrorMsg: line;
  begin
    if (TestClk'event and TestClk = '0') then
      if Qout /= testVector.Q then
        write(ErrorMsg, string'("Vector failed "));
        write(ErrorMsg, now);
        writeline(output, ErrorMsg);
      end if;
    end if;
  end process;


end testbench;
```