Review and follow the general instructions from lab 1.

In this lab, you’ll be designing and implementing a circuit that implements the classic 15-puzzle game (see picture at right). The objective of the puzzle is to slide the tiles around the board so as to produce the following arrangement of tiles.

```
_  1  2  3
4  5  6  7
8  9 10 11
12 13 14 15
```

Your fifteenPuzzle module should use the interface shown below.

```vhdl
entity fifteenPuzzle is port (  
    clk, reset: in std_logic;  
    -- client-side interface  
    nuPuzzle: in std_logic;  
    moveDir: in std_logic_vector(1 downto 0);  
    moveNow: in std_logic;  
    puzlTime, bestTime: out bcdVec(5 downto 0);  
    -- auxiliary signals for testing  
    peekPos: in nibble; -- specifies a board position  
    peekVal: out nibble; -- tile at position peekPos  
    emptyPos: out nibble; -- position of empty square  
    -- interface to external display  
    hSync, vSync: out std_logic;  
    dispVal: out pixel);  
end fifteenPuzzle;
```

The `nuPuzzle` input is raised high for one clock tick to make the circuit randomly scramble the tiles. The `moveNow` input is raised high for one clock tick to move one tile, based on the `moveDir` input. The `moveDir` input is a two bit signal that specifies one of the “neighbors” of the empty space on the board. `moveDir=0` refers to the tile above the empty space, `moveDir=1` refers to the tile to the right of the empty space, `moveDir=2` refers to the tile below the empty space and `moveDir=3` refers to the tile to the left of the empty space. When `moveNow` is asserted, the circuit should move the tile specified by `moveDir` into the empty space. Alternatively, we can think of each move as moving the empty space, in the direction specified by `moveDir`. The output `puzlTime` is a 6 digit BCD value that represents the time that has been spent solving the current puzzle. The units are hundredths of a second. The output `bestTime` is the smallest time for any puzzle solved so far. So whenever the user “solves” the puzzle, the value of `puzlTime` stops increasing, and it is compared to `bestTime`; if it is smaller, then `bestTime` is updated. The values of `puzlTime` and `bestTime` will be displayed on the LCD display, when using the prototype board.

There are three auxiliary signals that are useful for testing the circuit. The `peekPos` input specifies one of 16 “positions” in the array of tiles. Positions are numbered 0 through 15, starting in the
upper left corner and going from left-to-right along the first row, then the second row, and so forth. The `peekVal` output is the number on the tile at the position specified by `peekPos`. So for example, in the configuration shown in the picture above, if `peekPos=7`, then `peekVal` should equal 10. If the empty space is at the position specified by `peekPos`, then `peekVal` should be zero. Finally, the output `emptyPos` is the position on the game board where the empty square can be found. So for example, in the configuration shown in the picture, `emptyPos` would be 1.

The `commonDefs` file contains two functions, `plusOne` and `lessThan` that you will find useful for handling the two BCD values.

The `fifteenPuzzle` component instantiates a `copyPattern` component, similar to the one used in the `mineSweeper` game discussed in the text.

You will be required to complete the provided `outMod`, which displays the values of `puzlTime` and `bestTime`. See the `outMod` from lab 3, if you’re unsure about how to do this.

Your repository includes copies of all the source files you will need. You will find incomplete versions of `fifteenPuzzle`, `copyPattern`, `outMod`, `top` and `testTop` (among other things). Start by completing `fifteenPuzzle` and `copyPattern` and testing them using the provided testbench `testFifteen`. Then, proceed to complete `top`, `outMod` and `testTop`. Once your top level simulation is working correctly, test your circuit on the prototype board and see how quickly you can solve the puzzle. Post your best time on Piazza to claim bragging rights (sorry, no extra points).

For this lab, you may work with a partner. You are advised to select a partner whose performance in the course is roughly comparable to your own. It’s generally not a good idea to work with someone who doesn’t match your own level of understanding of the course material. When students with mismatched abilities do work together, the stronger partner usually ends up doing most of the work and the weaker partner learns very little from the experience. This often leads to bad feelings on both sides and undermines the educational value of the lab.

You only need to turn in a single lab report, but it should include the names of both partners, with one name circled. Commit the repository of the partner whose name is circled on the report.