Review and follow the general instructions from lab 1.

In this lab, you will be designing and implementing a `patternMatcher` circuit. This circuit will detect strings matching the regular expression `a*(b^n | c)d`. Here, the `+` symbol indicates one or more occurrences of the symbol `a`, the superscript `n` indicates exactly `n` repetitions of the symbol `b`. The value of `n` is fixed when the circuit initialized, so if `n=3` the circuit will match the strings `abbbd`, `aaacd`, `aabbbd` and so forth. The interface for the `patternMatcher` is defined by the following entity specification.

```vhdl
entity patternMatcher is port(
  clk, restart, valid: in std_logic;
  inSym: in nibble;
  repCount: in nibble;
  patCount: out byte);
end patternMatcher;
```

The `restart` signal re-initializes the circuit. The `valid` signal is high during any clock tick when there is a new input symbol. The `inSym` signal is a four bit signal that represents the input symbol, where `x0` corresponds to `a`, `x1` corresponds to `b` and so forth. The value of `inSym` is ignored whenever `valid` is low. `RepCount` determines the number of repetitions of the symbol `b` in the pattern. The circuit should store the value of `repCount` whenever `restart` is high, and use the stored value when matching patterns. The output `patCount` is the number of times an input string has matched the pattern since the last `restart`.

You should start by creating a state transition diagram for `patternMatcher`. Assign symbolic names to the required states and label all the state transitions with the input condition that causes the transition and any actions that should take place when the transition occurs. Check your state transition carefully to make sure you have covered all the cases. Then use it to guide you as you develop your VHDL specification of the circuit.

You will be required to simulate your circuit by itself using a testbench, `testPatMatch`, that you will find in your repository.

Next, you will be required to create a `top` circuit that combines your `patternMatcher` with provided input and output modules. The `restart` input of `patternMatcher` should be connected to output `pulse(1)` of the `binaryInMod` circuit, `valid` should be connected to `pulse(2)`, `inSym` should be connected to `inBits(3..0)` and `repCount` should be connected to `inBits(11..8)`. The `outMod` is a new circuit designed to be used with the `patternMatcher`. Its `valid` and `inSym` inputs should be connected to the corresponding inputs of `patternMatcher` and its `patCount` input should be connected to the `patCount` of `patternMatcher`. You should also connect the `inSym` and `repCount` inputs of `patternMatcher` to `led(3..0)` and `led(7..4)` respectively. The `outMod` circuit displays the current `inSym` value as an ASCII character at the right end of the top row of the display. Recently input symbols appear to the left of the current symbol. The value of `patCount` is displayed in hex in the bottom row of the display.
The next step is to complete a partial testbench called *testTop* that you will find in your repository. Note that *testTop* defines a number of helper procedures to make this easier. Read the code and try to understand how they work. Ask questions if you’re not sure. Then, add tests to the testbench, following the instructions you will find in the provided file. Use your testbench to simulate the complete circuit. Be sure to include all internal signals of your *patternMatcher* in the waveform display (do not forget the *state* signal). Also include *outMod’s* internal signals, *top* and *bot*. These show the characters that will appear on the external LCD display. Check the simulation output carefully to verify that all components are connected correctly and that your circuit is producing the correct output.

The last step in this lab is to load your circuit onto a prototype board and demonstrate to a TA that it works correctly. Procedures for signing out a board will be announced in class and posted on the web site. You should resist the temptation to test on the prototype board until you have carefully checked the simulation of your completed circuit. Use the simulator to get the bugs out. If you do this, the prototype testing should be quick and easy. If you don’t, you’re likely to find yourself wasting a lot of time. It’s nearly impossible to find design errors using the prototype board. The simulator is a much more powerful tool for that purpose, since it allows you to see what’s happening inside your circuit. You will find more details in the lab report template.

*For this lab, you are expected to work as individuals. While you can discuss the lab in general terms with your classmates, you are not permitted to collaborate in any way. Sharing of VHDL code, diagrams, design notes or simulation results is explicitly prohibited. Students found violating these rules will have the full value of this lab deducted from their class scores.*