PROBLEM 1: GENERAL QUESTIONS
Answer the following questions in a concise way. You are encouraged to look for the answer online. These questions will not be graded for the homework and you are not required to provide a report for them or submit anything, but they might be part of the final exam.

1) What is the difference between wire and reg?
2) What is the difference between blocking and nonblocking assignments?
3) What is the difference between bit wise, unary and logical operators?
4) Write a Verilog code to swap two registers without using a temporary register.
5) What is the difference between posedge and negedge keywords?
6) What is the difference between $display and $monitor?
7) Given the following Verilog code, what value of “a” is displayed?

```verilog
testbench: always @(clk) begin
  a = 0;
  a <= 1;
  $display(a);
end
```

8) What is the difference between the following two lines of Verilog code?

```verilog
#5 a = b;
a = #5 b;
```

9) What is the difference between the following lines of code?

```verilog
reg1<= #10 reg2;
reg3 = # 10 reg4;
```

10) What is the difference between === and == in Verilog?

PROBLEM 2: RIPPLE CARRY ADDER
Using full adders as building blocks, implement a 4-bit adder in the ripple carry form. Use explicit module connection.

Do not use behavioral modeling for this problem. (e.g. no “always” or “initial” blocks).

Write a testbench to print the following results with numbers represented in both decimal and binary format:
- 3 + 5
- 7 + 7
- 9 + 6
- 10 + 10
**PROBLEM 3: CARRY LOOKAHEAD ADDER**
Using full adders as building blocks, implement a 4-bit adder in the **carry lookahead** form.
Use explicit module connection.
Do not use behavioral modeling for this problem. (e.g. no “always” blocks)
Write a testbench to print the following results with numbers represented in both decimal and binary format:
- 3 + 5
- 7 + 7
- 9 + 6
- 10 + 10

**PROBLEM 4: MULTIPLEXER**
Using 2:1 MUX as a building block, implement a 8:1 MUX
Use explicit module connection.

**PROBLEM 5: ARITHMETIC LOGIC UNIT (ALU)**
Implement an ALU with control signals S0, S1 that performs the following operations with 8-bit input operands A, B, C.
**Hint: use behavioral modeling**

<table>
<thead>
<tr>
<th>Operation ID</th>
<th>S0</th>
<th>S1</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A*B</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>A-C</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>(A xnor B) xnor C</td>
</tr>
</tbody>
</table>
| 3            | 1  | 1  | If A[0] == 1
                |    |    |   B+C                      |
                |    |    | Else
                |    |    |   B-C                      |

Write a test bench to report the output of the operations with the following values for the input operands (expressed in base ten in the following table).

<table>
<thead>
<tr>
<th>Operation ID</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>20</td>
<td>34</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>34</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>255</td>
<td>128</td>
</tr>
</tbody>
</table>
PROBLEM 6: THE FIRST SYNCHRONOUS DESIGN
Design and implement a counter that starts at 15 and counts down to 0. There are several inputs to this circuit.
[start] : start the counter
[stop] : stop the counter
[reset] : reset back to 15 seconds
The output will include:
[4-bit counter] (in binary is OK)
[alarm]: indicator that switches from 0 to 1 when the sequence reaches 0000.
Once the counter reaches 0, the alarm indicator should stay at 1 until the counter is reset.
Simulate with start=0 first, then switch start=1. The counter should be at 1111 & the alarm indicator should be 0 until the counter becomes 0000. Also, simulate the case that when start=1 but you realize at 6 second before the alarm sounds, you need to stop it & reset it back to 15 seconds again.

Checkoff list (to be completed with a tutor, a TA or the instructor before the end of last available office hour on Wednesday 08/17, which starts at 3:30PM).
NOTE: before starting the checkoff, a student MUST submit its zipped project via TED.
The submitted material MUST contain the source code.
The checkoff can only be attempted once, so make sure that your solution is working fine.
You can have your projects compiled in advance.
  1) Submit the zipped project folder containing ALL of your source code on TED
  2) Explain the functionality of 2 modules (problems from 2 to 6), selected by the TA

For each of the two modules:
  3) Review the Verilog code
  4) Run testbench and explain the results to verify the functionality