Review and follow the general instructions from lab 1.

In this lab, you’ll be designing and implementing a stack-based calculator. There will be two parts to this circuit, a general stack component, and a stackCalc that uses the stack to store values. The stack component should use the following entity declaration.

```vhdl
entity stack is
  generic(
    stackSiz: integer := 8;
    lgSiz:   integer := 3;
    wordSiz: integer := 16);
  port(
    clk, reset: in std_logic;
    push, pop: in std_logic;
    dIn: in std_logic_vector(wordSiz-1 downto 0);
    top: out std_logic_vector(wordSiz-1 downto 0);
    empty, full: out std_logic);
end stack;
```

This entity uses three generic parameters. StackSiz is a power of 2 that is one larger than the number of elements that the stack can hold. LgSiz is the base 2 logarithm of StackSiz. WordSiz is the number of bits in the data values stored in the stack. The input signal push is used to add a new value to the stack. The output signal pop is used to remove the top value from the stack. When push and pop are high at the same time, the input value dIn replaces the value on the top of the stack. The output top is the value of the top element of the stack, whenever the stack is not empty. The outputs empty and full indicate if the stack is empty or full. A push on a full stack should be ignored. A pop on an empty stack should also be ignored. Also, if push and pop are both high when the stack is empty, the operation should be ignored.

You will write a testbench to verify the operation of your stack component. This testbench should instantiate a stack with a stackSiz parameter equal to 8 and a wordSiz parameter equal to 16. Your testbench should fill the stack with unique values and empty it. Your testbench should use assertions to verify that the value on the top is correct at all times and that the full and empty signals are correct.

The stackCalc component should use the following entity specification.

```vhdl
entity stackCalc is
  generic(
    stackSiz: integer := 8;
    lgSiz:   integer := 3;
    wordSiz: integer := 16);
  port(
    clk, reset: in std_logic;
    op: in nibble;
    doOp: in std_logic;
    dIn: in std_logic_vector(wordSiz-1 downto 0);
```

```
result: out std_logic_vector(wordSiz-1 downto 0));
end stackCalc;

The *op* input specifies one of eight operations, specified below.

0. Clear the value on the top of the stack.
1. Clear the entire stack.
2. Pop a value from the stack.
3. Push the input value *dIn* onto the stack.
4. Add the input value *dIn* to the value on the top of the stack and replace the value on the top of the stack with the sum.
5. Add the top two values on the stack together and replace them both with their sum.
6. Subtract *dIn* from the value on the top of the stack and replace the value on the top of the stack with the difference.
7. Subtract the top two values on the stack and replace them both with their difference.

The *doOp* input controls when an operation occurs. It should be high for one clock tick in order to initiate an operation. Most operations should complete in a single clock tick. Operations 5 and 7 should take two clock ticks to complete. If the stack is empty when an add/subtract operation is performed, the requested operation should be ignored. Also, if the stack has one element when operations 5 or 7 are requested, the operation should not be performed, and the stack returned to its original state.

You will be provided with a testbench for the *stackCalc* component. You will also be provided with a top circuit and supporting components that you can use to test your circuit on the prototype board.

You will find more detailed instructions in the lab report template in your repository.

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.