```
module fourBitAdder(a,b,s,co);
input wire[3:0]a,b; //declare two four bit inputs a and b
output wire [3:0]s; //declare one four bit output s
output wire co;     //a one bit output for the carry out

wire co0,co1,co2; //declare three intermediate wires that
                  //will connect the output of one module
                  //to the input of another module.
/*
SYNTAX NOTE:
if I have a module that has the following declaration
      testModule(in1,in2,out1);//all 1 bit width
and the following code in another module:
      wire a,b,c;
      testModule one(.in1(a),.in2(b),.out1(c));

it will create a module that uses a for the input to in1
b for the input to in2
c for the output of out1

meaning the syntax is:
   .port_on_module_you_are_connecting_to(wire_in_current_module)


*/


//create our first half adder co0 is a output for carryout that will
//serve as the carryin for the first fulladder(fa1)
halfAdder ha0(.a(a[0]),.b(b[0]),.s(s[0]),.co(co0));



//create the first full adder(fa1) that takes in co0 as the carry in bit
//and has co1 as the carryoutbit which feeds intofa2
fullAdder fa1(.a(a[1]),.b(b[1]),.ci(co0),.co(co1),.s(s[1]));

//create the second full adder(fa2) that takes in co1 as the carry
//in bit from fa1 and has co2 as teh carry out bit which feeds into fa3

fullAdder fa2(.a(a[2]),.b(b[2]),.ci(co1),.co(co2),.s(s[2]));


//create the last full adder(fa3) that takes in co2 as its carry in bit
//from fa2 and then outputs the final sum bit and the fourBitAdders carryout
fullAdder fa3(.a(a[3]),.b(b[3]),.ci(co2),.co(co),.s(s[3]));


endmodule


module halfAdder(a,b,s,co);//halfAdder declaration
input wire a,b;//input of a and b
output wire s,co;//outputs

//output logic that is found in your book!
assign s = a^b;
assign co = a&b;

endmodule

module fullAdder(a,b,ci, co, s);//fullAdder declaration
input wire a, b, ci;//full adder also has a carry in bit
output wire co, s; //outputs

//output logic that is found in your book!
assign s = a^b^ci;
assign co = (b&ci)|(a&ci)|(a&b);
endmodule
```