6.004 Recitation 8
L08 - More on Combinational Circuits

Problem 1.

Write a Bluespec function checkParity, which takes as input a 5-bit vector and returns True if the input vector has an odd number of 1's, otherwise it returns a false.

Also, think about the propagation delay of your solution.

```
function Bit#(1) checkParity(Bit#(5) in);
    return in[4] ^ in[3] ^ in[2] ^ in[1] ^ in[0];
endfunction
```

Problem 2.

Write Bluespec function addParity, which takes as input a 4 bit vector and returns a 5 bit vector which adds an odd-parity bit to the input in the most significant position.

Note that an odd-parity bit makes the number of 1's in the resulting bit vector odd and an even-parity bit makes the number of 1's in the resulting bit vector even.

```
function Bit#(5) addParity(Bit#(4) in);
    // checks if input has odd number of ones
    Bit#(1) oddNumOnes = in[3] ^ in[2] ^ in[1] ^ in[0];
    // if yes, add 0, if not, add 1 to make odd-parity
    Bit#(1) parity = (oddNumOnes == 1)? 0: 1;
    return {parity, in};
endfunction
```

Problem 3.

In Lab 3, we wrote a function that computed whether a 4-bit input is a power of 2 or not (basically, this was checking if there is only one bit in the input which is equal to 1). Rewrite this function so that it works with inputs of arbitrary bit width.

```
function Bit#(1) isPowerOfTwo(Bit#(n) in);
      Bit#(1) out = 0;
      Bit#(2) count = 0;
      for (Integer i = 0; i<valueOf(n); i=i+1) begin
            count = (in[i]==1)?(count+1):(count);
      out = (count==1)? 1 : 0;
      return out;
endfunction
```

Problem 4:

Parameterize the bit-scan-reverse function from Lab3 to take as input a w-bit vector and output the index of the first non-zero bit scanned from the largest index.

```
function Bit#(TLog#(w)) bitScanReverse (Bit#(w) in);
      Bit#(TLog#(w)) ret = 0;
      for (Integer i=0; i<valueOf(w); i=i+1) begin
            ret = (in[i] == 1) ? fromInteger(i) : ret;
      end
      return ret;
endfunction
```

Note: TLog#(w) returns a type of ceiling(log2(w))