Arithmetic

Part A For each problem, (a) compute the operation using the rules of addition, and (b) indicate whether an overflow occurs. All numbers are expressed using a **six** bit, two's compliment representation in binary notation.

1 0 1 1	0 1 0 0 0 0	1 0 1 0 1 1	0 1 1 1 1 1
+ 1 1 1 0	+ 1 1 1 1 1 1	+ 1 1 0 1 0 1	+ 0 0 0 0 0 1
overflow? _____	overflow? _____	overflow? _____	overflow? _____

1 1 0 1 1	0 1 0 1 0 0	1 1 1 0 1 1
+ 1 1 1 1	+ 1 1 1 1 1 1	+ 1 1 0 0 0 1
overflow? _____	overflow? _____	overflow? _____

1 1 1 1	1 1 1 0 0 0	1 1 0 0 0 0
+ 1 1 1 1	+ 0 0 1 1 1 1	+ 1 1 1 1 1 1
overflow? _____	overflow? _____	overflow? _____

Part B Now compute additions and detect overflows for these **five** bit, two's compliment numbers.

1 0 1 1	1 0 1 0 0	1 0 1 0 1	1 1 1 1
+ 1 0 1	+ 1 0 1 0 1	+ 1 0 1 1	+ 1 0 0 0 0
overflow? _____	overflow? _____	overflow? _____	overflow? _____

Part C For each problem, (a) compute the operation using the rules of addition, (b) indicate whether an error occurs assuming all numbers are expressed using a **five** bit, two's compliment representation, and (c) indicate whether an error occurs assuming all numbers are expressed using a **five** bit, unsigned binary representation. All number are in binary notation.

1 1 0 1	1 0 1 1 0	1 1 0 1 0	1 0 1
+ 1 1 1	+ 1 1 0 1 1	+ 1 0 1 0 1	+ 1 0 0 1
signed error? _____	signed error? _____	signed error? _____	signed error? _____
unsigned error? _____	unsigned error? _____	unsigned error? _____	unsigned error? _____

1 1 1 1 1	1 1 1 1	1 0 0 0 1	1 0 0 0 1
+ 1 1 1 1 1	+ 1 1 1 1	+ 1 1 1 1	+ 1 0 0 0 0
signed error? _____	signed error? _____	signed error? _____	signed error? _____
unsigned error? _____	unsigned error? _____	unsigned error? _____	unsigned error? _____

Part D For each problem, compute the subtraction by negating the second operand and then adding the binary operands. All numbers are expressed using a **six** bit, two's compliment representation in binary notation. Don't worry about overflows here.

0 1 1 0 0 1	0 1 0 1 1 0	0 1 1 1 1 1
- 0 0 1 1 1 1	- 0 1 0 1 0 1	- 0 1 1 1 1 1

Part E Now compute these subtractions (by negating the second operand and adding) for these **five** bit, two's compliment numbers. Again, don't worry about overflows.

1 0 0 1	1 0 1 0	1 0 1 0 1	1 1 0 0 1
- 1 1 0 0	- 1 0 1	- 1 0 1 0 1	- 1 0 0 1 1

1 1 1 0	1 0 1	1 0 1 0 1	1 1 1 1 1
- 1 1 1 1 1	- 1 1 1	- 1 0 1 1 0	- 1 0 0 0 0

1 1 1 0	1	1 1 0 1 1	1 1 0 0 1
- 1 1 1 0 0	- 1 0	- 1 1 1	- 1 1 0 0 1

Part F Two eight bit two's compliment numbers are added together to form an eight bit two's compliment result. The inputs are named X7:X0 and Y7:Y0. The result is named Z7:Z0. Assume other circuitry (an eight bit adder) computes Z from X and Y. Your task is to design the logic that detects when an addition overflow (positive or negative) occurs using any bits from X, Y, and Z as **inputs**. Your overflow detector should have one output, OVERFLOW, that is high when an overflow is detected.