ADDITION IN BINARY WITH LOGICAL OPERATORS

Truth Table

p	q	CarryIn	Sum	CarryOut
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

+ 10101100 <--Addend-One (p)
+ 00110011 <--Addend-Two (q)
= 11011111

1. **Move within the p and q strings bit-by-bit from right to left, setting Integer Variables for each bit as you go.**
2. **Start with CarryIn = 0 and CarryOut = 0**
3. **Add Addend-One (p) to Addend-Two (q), bit-by-bit as shown below.**
4. **The Sum is calculated bit-by-bit from right to left as follows:**

   Sum = (p XOR q) XOR CarryIn
5. **Store a "running sum" in a Concatenated String from right to left:**

   Sum$ = [String Digit of Sum] + Sum$
6. **The CarryOut is calculated bit-by-bit as follows:**

   CarryOut = (p AND q) OR (p AND CarryIn) OR (q AND CarryIn)
7. **CarryIn = CarryOut**
8. **CarryOut = 0**
9. **Continue through the p and q strings bit-by-bit until done (Go back to 4.)**
10. **If there is a final CarryOut, then concatenate to Sum$: Sum$ = [String Digit 1] + Sum$**

SUBTRACTION IN BINARY WITHOUT BORROWING
(USING LOGICAL OPERATORS)

```
+ 11010011  <--Minuend
- 10011100  <--Subtrahend
= 00110111
```

To keep from borrowing, use the ONE'S COMPLEMENT as follows:

(a) + 11010011 - 10011100 (+ 11111111 + 1 - 100000000)	The binary numbers in the parenthesis add to ZERO
(b) + 11010011 (+ 11111111 - 10011100) +1 - 100000000	Rearrange terms for NO borrowing
(c) + 11010011 (+ 01100011) + 1 - 100000000	Subtract in the parenthesis (SAME as Subtrahend XOR'd with 1's)
(d) (+ 11010011 + 01100011) + 1 -100000000	Rearrange terms within the parenthesis
(e) (+ 100110110) + 1 - 100000000	Add in the parenthesis
(f) (+ 100110110 + 1) - 100000000	Rearrange terms within the parenthesis
(g) (+ 100110111) - 100000000	Add in the parenthesis
(h) + 100110111 - 100000000 = 00110111	Same answer WITHOUT borrowing

Therefore for Binary Subtraction:

1. Make sure the Minuend String is LARGER than the Subtrahend String. If NOT, SWAP them and set flag for NEGATIVE result.
2. Move within the Minuend and Subtrahend strings from right to left, setting Integer Variables for each bit as you go.
3. Start with CarryOut = 0 and CarryIn = 1 (See (a) above)
4. Add the Minuend to the (Subtrahend XOR'd with 1), bit-by-bit, as shown below (See (c) above)
5. The Sum is calculated bit-by-bit as follows: Sum = (Minuend XOR (Subtrahend XOR 1)) XOR CarryIn
6. Store a "running sum" in a Concatenated String from right to left: Sum$ = [String Digit of Sum] + Sum$
7. The CarryOut is calculated bit-by-bit as follows:
   CarrryOut = (Minuend AND (Subtrahend XOR 1)) OR (Minuend AND CarryIn) OR ((Subtrahend XOR 1) AND CarryIn)
8. CarryIn = CarryOut
9. CarryOut = 0
10. Continue through the Minuend and Subtrahend Strings bit-by-bit until done (Go back to 5.)
11. There will ALWAYS be a final CarryOut: Ignore it or set it to zero (See (h) above: + 100110111 - 100000000)

SUBTRACTION IN BASE-10 WITHOUT BORROWING
(To help illustrate binary subtraction without borrowing)

```
 17522  ß-- Minuend
-16859  ß-- Subtrahend
   663
```

To keep from borrowing, use the NINE'S COMPLEMENT AS FOLLOWS:

(a) 17522 – 16859 + (99999 + 1 – 100000)	**The numbers in the parenthesis add to ZERO**
(b) 17522 + (99999 – 16859) + 1 – 100000	**Rearrange terms for NO BORROWING**
(c) 17522 + (83140) + 1 – 100000	**Subtract within the parenthesis**
(d) (17522 + 83140) + 1 – 10000	**Rearrange terms within the parenthesis**
(e) (100662) + 1 – 100000	**Add within the parenthesis**
(f) (100662 +1) – 100000	**Rearrange terms within the parenthesis**
(g) (100663) – 100000	**Add within the parenthesis**
(h) 100663 – 100000 = 663	**SAME answer WITHOUT borrowing**

Note: The number of NINE'S used is always the number of DIGITS in the Subtrahend.