1) Perform the following operations in binary and give the 8-bit binary result. (Ignore any carries.) Convert each number (including the 8-bit result) to decimal assuming both unsigned and signed. Circle each decimal result that is incorrect. The first is done as an example.

<table>
<thead>
<tr>
<th></th>
<th>unsigned</th>
<th>signed</th>
<th>unsigned</th>
<th>signed</th>
<th>unsigned</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>254</td>
<td>11111110</td>
<td>-2</td>
<td>01111110</td>
<td>01111111</td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td>255</td>
<td>+11111111</td>
<td>-1</td>
<td>+10000000</td>
<td>+01111100</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>253</td>
<td>11111101</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(d)</td>
<td>11110000</td>
<td>10111000</td>
<td>00011101</td>
<td>00110101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e)</td>
<td>+00000011</td>
<td>+01011101</td>
<td>+01101011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(f)</td>
<td>11110011</td>
<td>10101010</td>
<td>11100110</td>
<td>11100110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(g)</td>
<td>-10101010</td>
<td>-11110011</td>
<td>-01110101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(h)</td>
<td>00110011</td>
<td>01110111</td>
<td>11001010</td>
<td>11001010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(i)</td>
<td>01010101</td>
<td>01010101</td>
<td>01101010</td>
<td>01101010</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2) Perform the following 8-bit by 8-bit multiplications in binary and give the 16-bit result. Assume the numbers are unsigned. The first answer is given so you can check your method.

<table>
<thead>
<tr>
<th></th>
<th>unsigned</th>
<th>signed</th>
<th>unsigned</th>
<th>signed</th>
<th>unsigned</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>00110011</td>
<td>01110111</td>
<td>11001010</td>
<td>11001010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td>×01010101</td>
<td>×01010101</td>
<td>×01101010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00110011</td>
<td>00000000</td>
<td>00110011</td>
<td>00000000</td>
<td>00110011</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td>00110011</td>
<td>00000000</td>
<td>00110011</td>
<td>00000000</td>
<td>00110011</td>
</tr>
<tr>
<td></td>
<td>0001000011110111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3) on back side
3) Given the following

\[
x = 0b00111101; \quad \text{ // assumed signed values}
\]
\[
y = 0b10001110;
\]

perform each operation and give the 8-bit binary result to the left of each problem. The first is already done as an example. Multiplication, division and modulo can be converted to decimal before performing, but all other operations should be done in binary. Show any work required.

```
00111110  x + 1
x + y
y - x
x - y
x * 2
x / 8
x % 8
x / 10
x % 10
+x
-x
-y
~x
~y
!x
x << 2
y << 2
x >> 2
y >> 2
x & y
x | y
x ^ y
1 && 0
x && 0
x && 1
x && y
x || 1
x || y
1 == 0
x == 61
x == y
x != 12
x != y
x < y
x <= y
x > y
x >= y
```