Homework Assignment 4 Answered

Logical Shift Right

No work needed.

1. 0110 by 3

```
0110 >> 1 => 0011

0110 >> 2 => 0001

0110 >> 3 => 0000
```

2. 1110 by 1

```
1110 >> 1 => 0111
```

3. 001011 by 7

```
001011 >> 1 => 000101

001011 >> 2 => 000010

001011 >> 3 => 000001

001011 >> 4 => 000000

001011 >> 5 => 000000

001011 >> 6 => 000000

001011 >> 7 => 000000
```

4. 111111 by 4

```
111111 >> 1 => 011111

111111 >> 2 => 001111

111111 >> 3 => 000111

111111 >> 4 => 000011
```

5. 01111111 by 2

```
01111111 >> 1 => 00111111

01111111 >> 2 => 00011111
```

6. 10000000 by 6

```
10000000 >> 1 => 01000000

10000000 >> 2 => 00100000

10000000 >> 3 => 00010000

10000000 >> 4 => 00001000

10000000 >> 5 => 00000100

10000000 >> 6 => 00000010
```

7. 000000000101 by 1

```
000000000101 >> 1 => 000000000010
```

8. 111111111111 by 9

```
111111111111 >> 1 => 011111111111

111111111111 >> 2 => 001111111111

111111111111 >> 3 => 000111111111

111111111111 >> 4 => 000011111111

111111111111 >> 5 => 000001111111

111111111111 >> 6 => 000000111111

111111111111 >> 7 => 000000011111

111111111111 >> 8 => 000000001111

111111111111 >> 9 => 000000000111
```

9. 101010 by 0

```
101010 >> 0 => 101010
```

10. 0101 by 3

```
0101 >> 1 => 0010

0101 >> 2 => 0001

0101 >> 3 => 0000
```

Arithmetic Shift Right

No work needed.

1. 0110 by 3

```
0110 >> 1 => 0011

0110 >> 2 => 0001

0110 >> 3 => 0000
```

2. 1110 by 1

```
1110 >> 1 => 1111
```

3. 001011 by 7

```
001011 >> 1 => 000101

001011 >> 2 => 000010

001011 >> 3 => 000001

001011 >> 4 => 000000

001011 >> 5 => 000000

001011 >> 6 => 000000

001011 >> 7 => 000000
```

4. 111111 by 4

```
111111 >> 1 => 111111

111111 >> 2 => 111111

111111 >> 3 => 111111

111111 >> 4 => 111111
```

5. 01111111 by 2

```
01111111 >> 1 => 00111111

01111111 >> 2 => 00011111
```

6. 10000000 by 6

```
10000000 >> 1 => 11000000

10000000 >> 2 => 11100000

10000000 >> 3 => 11110000

10000000 >> 4 => 11111000

10000000 >> 5 => 11111100

10000000 >> 6 => 11111110
```

7. 000000000101 by 1

```
000000000101 >> 1 => 000000000010
```

8. 111111111111 by 9

```
111111111111 >> 1 => 111111111111

111111111111 >> 2 => 111111111111

111111111111 >> 3 => 111111111111

111111111111 >> 4 => 111111111111

111111111111 >> 5 => 111111111111

111111111111 >> 6 => 111111111111

111111111111 >> 7 => 111111111111

111111111111 >> 8 => 111111111111

111111111111 >> 9 => 111111111111
```

9. 101010 by 0

```
101010 >> 0 => 101010
```

10. 0101 by 3

```
0101 >> 1 => 0010

0101 >> 2 => 0001

0101 >> 3 => 0000
```

Logical Shift Left

No work needed.

1. 0110 by 3

```
0110 << 1 => 1100

0110 << 2 => 1000

0110 << 3 => 0000
```

2. 1110 by 1

```
1110 << 1 => 1100
```

3. 001011 by 7

```
001011 << 1 => 010110

001011 << 2 => 101100

001011 << 3 => 011000

001011 << 4 => 110000

001011 << 5 => 100000

001011 << 6 => 000000

001011 << 7 => 000000
```

4. 111111 by 4

```
111111 << 1 => 111110

111111 << 2 => 111100

111111 << 3 => 111000

111111 << 4 => 110000
```

5. 01111111 by 2

```
01111111 << 1 => 11111110

01111111 << 2 => 11111100
```

6. 10000000 by 6

```
10000000 << 1 => 00000000

10000000 << 2 => 00000000

10000000 << 3 => 00000000

10000000 << 4 => 00000000

10000000 << 5 => 00000000

10000000 << 6 => 00000000
```

7. 000000000101 by 1

```
000000000101 << 1 => 000000001010
```

8. 111111111111 by 9

```
111111111111 << 1 => 111111111110

111111111111 << 2 => 111111111100

111111111111 << 3 => 111111111000

111111111111 << 4 => 111111110000

111111111111 << 5 => 111111100000

111111111111 << 6 => 111111000000

111111111111 << 7 => 111110000000

111111111111 << 8 => 111100000000

111111111111 << 9 => 111000000000
```

9. 101010 by 0

```
101010 << 0 => 101010
```

10. 0101 by 3

```
0101 << 1 => 1010

0101 << 2 => 0100

0101 << 3 => 1000
```

Arithmetic Shift Left

No work needed.

1. 0110 by 3

```
0110 << 1 => 1100

0110 << 2 => 1000

0110 << 3 => 0000
```

2. 1110 by 1

```
1110 << 1 => 1100
```

3. 001011 by 7

```
001011 << 1 => 010110

001011 << 2 => 101100

001011 << 3 => 011000

001011 << 4 => 110000

001011 << 5 => 100000

001011 << 6 => 000000

001011 << 7 => 000000
```

4. 111111 by 4

```
111111 << 1 => 111110

111111 << 2 => 111100

111111 << 3 => 111000

111111 << 4 => 110000
```

5. 01111111 by 2

```
01111111 << 1 => 11111110

01111111 << 2 => 11111100
```

6. 10000000 by 6

```
10000000 << 1 => 00000000

10000000 << 2 => 00000000

10000000 << 3 => 00000000

10000000 << 4 => 00000000

10000000 << 5 => 00000000

10000000 << 6 => 00000000
```

7. 000000000101 by 1

```
000000000101 << 1 => 000000001010
```

8. 111111111111 by 9

```
111111111111 << 1 => 111111111110

111111111111 << 2 => 111111111100

111111111111 << 3 => 111111111000

111111111111 << 4 => 111111110000

111111111111 << 5 => 111111100000

111111111111 << 6 => 111111000000

111111111111 << 7 => 111110000000

111111111111 << 8 => 111100000000

111111111111 << 9 => 111000000000
```

9. 101010 by 0

```
101010 << 0 => 101010
```

10. 0101 by 3

```
0101 << 1 => 1010

0101 << 2 => 0100

0101 << 3 => 1000
```

AND, OR, XOR Functions

What would be the results of AND, OR, and XOR. Treat each bit of the following binary numbers as individual inputs to each gate. No work needed.

Example: 1001001 => AND = 0, OR = 1, XOR = 1

1. 0110 => AND = 0, OR = 1, XOR = 0
2. 1110 => AND = 0, OR = 1, XOR = 1
3. 001011 => AND = 0, OR = 1, XOR = 0
4. 111111 => AND = 1, OR = 1, XOR = 0
5. 01111111 => AND = 0, OR = 1, XOR = 1
6. 10000000 => AND = 0, OR = 1, XOR = 1
7. 000000000101 => AND = 0, OR = 1, XOR = 0
8. 111111111111 => AND = 1, OR = 1, XOR = 0
9. 101010 => AND = 0, OR = 1, XOR = 1
10. 0101 => AND = 0, OR = 1, XOR = 0