Week7-Lab - Solution

1.

```
8 & 3 = 0
```

```
(5 | 3) | 3 = 7
```

```
5 | 3 = 7
```

```
(5 ^ 3) ^ 3 = 5
```

```
5 ^ 3 = 6
```

2. Let the bit pattern for x be $b_8b_7b_6b_5b_4b_3b_2b_1$. The bit pattern for 80_{16} is 10000000.

So `x & 0x80` gives &

b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1
1	0	0	0	0	0	0	0
b_8	0	0	0	0	0	0	0

So `(x & 0x80) >> 7)` is b_8

`x << 1` is $b_8b_7b_6b_5\,b_4b_3b_2b_10$

So `(x << 1) & 0xFF` is &

b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	0
0	1	1	1	1	1	1	1	1
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	0

So `((x & 0x80) >> 7) ^ ((x << 1) & 0xFF)` is $\oplus$

b_7	b_6	b_5	b_4	b_3	b_2	b_1	0
0	0	0	0	0	0	0	b_8
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_8

So the result is a circular shift left of the bits in x.

By a similar argument `((x & 0x01) << 7) ^ ((x >> 1) & 0x7F)` results in a circular shift right of the bits in x.

3.

```
ord('A') << 2 = 260
```

```
ord('A') & 3 = 1
```

```
ord('A') >> 1 = 32
```

```
ord('A') | 3 = 67
```

```
ord('A') >> 2 = 16
```

```
ord('A') ^ 3 = 66
```

```
ord('A') >> 2) << 1 = 32
```

```
chr(ord('A')^3) = 'B'
```

```
ord('C') >> 2) << 1 = 32
```

```
chr((ord('A')^23)^23) = 'A'
```

4. The second program. << is more "efficient" than pow().

5. To view help on the *baseconvert* function in the byte complied module *baseconv.pyc* type *help()* at the *idle* prompt then type *baseconv.baseconvert*

```
# asciibin3.py -- reads a printable character and
# outputs its decimal and binary ascii code
from baseconv import *
character = raw_input('Input a printable character: ')
bits = baseconvert(ord(character),BASE10,BASE2)
print 'The character %c has a decimal ascii code of %d\n\
and a 7 bit binary code of %s' % (character, ord(character), bits)
```

Here is yet another way of achieving the same result.

```
# asciibin4.py -- reads a printable character and
# outputs its decimal and binary ascii code

import sys

def bin(i):
    j=0
    if(i!=0):
        j=i
        bin(i>>1)
        sys.stdout.write(j&1)

character = raw_input('Input a printable character: ')
acharacter = ord(character)
print 'The character %c has a decimal value of %d' % (character,acharacter)
print 'and a binary code of ',; bin(acharacter)
```

Here is another script in similar vein. What does it do? Can you explain how it does it?

```
# mystery.py

import sys

mask = 0x40
char = raw_input('Input a printable character: ')
byte = ord(char)
sys.stdout.write('%c %d %x' % (char,byte,byte))
for i in range(7):
    sys.stdout.write(((mask >> i) & byte) >> (6-i))
```