1.

Assume:

```
int x = rand();
int y = rand();
unsigned ux = (unsigned) x;
```

Are the following statements always true?

a.

```
ux >> 3 == ux/8
```

True, right shifting a non-negative value yields the same result as dividing. Only when right shifting a negative value will right shifting round differently than from dividing.

b.

given x > 0,

```
((x << 5) >> 6) > 0
```

False. If the result of left shifting x by 5 has a 1 as the MSB, then the answer will ultimately be less than 0, because after right shifting by 6, if arithmetic shift is used, then the answer will still be less than 0.

c.

```
~x + x >= ux
```

True. ~x + x is always -1, and after casting that to an unsigned int, it is Umax.

d.

given x & 15 == 11,

```
( ~((x >> 3) & (x >> 2)) << 31) >= 0
```

False. The result can be determined through bitwise computation, and for the result, the MSB is 1. Thus, left shifting that by 31 means the MSB will be 1 afterwards, which is a negative value.

e.

given ((x < 0) && (x + x < 0))

```
x + ux < 0
```

False. x + ux results in an unsigned result because of implicit casting, regardless of what x contains beforehand.

f.

given ((x < 0) && (y < 0) && (x + y > 0))
((x | y) >> 30) == -1

If the resulting statement is to be true, then x | y must have a 1 in the MSB position and a 1 to the right of that. Thus, an arithmetic right shift will result in all ones. [x | y = 11-----…-] I would say this does not have to be true if the previous conditionals are all true, because if x is T_{min} and y is T_{min} + 1, x < 0 is true, y < 0 is true, and x + y > 0 is true.

[proof of $T_{min} + T_{min} + 1 > 0$]

$$(T_{min} + T_{min} + 1) \bmod 2^w = (-2^{w-1} - 2^{w-1} + 1) \bmod 2^w$$
$$= (2 * (-2^{w-1}) + 1) \bmod 2^w$$
$$= (-2^w + 1) \bmod 2^w$$
$$= 1$$

Thus, if x and y are T_{min} and T_{min} + 1, then the two MSBs of x | y will not both be 1, meaning the statement is false.

2.
Given: x has a 4 byte value of 66583 (dec_to_hex(x) = 0x00010417)
What is the value of the byte with the lowest address in an

a. big endian system? MSB at lowest address, LSB at highest address.

32-bit architecture

Address (0x)	100	101	102	103
Content (0x)	00	01	04	17

64-bit architecture

Address (0x)	100	101	102	103	104	105	106	107
Content (0x)	00	00	00	00	00	01	04	17

b. little endian system? MSB at highest address, LSB at lowest address

32 bit architecture

Address (0x)	100	101	102	103
Content (0x)	17	04	01	00

64 bit architecture

Address (0x)	100	101	102	103	104	105	106	107
Content (0x)	17	04	01	00	00	00	00	00