1.

Assume:

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

Are the following statements always true?

a.

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

True

- For unsigned integers, right shifting always rounds towards 0, as all unsigned integers are non-negative and extra 1's on the right are discarded while right shifting.
- Thus, shifting to the right by 3 is equivalent to integer division by 2^3, which also rounds towards 0.

b.

given x > 0,

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

False

- In the case where (x << 5) has a 1 for its most significant bit, right shifting by 6 will produce a negative number.

c.

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

True

- ~x + x would be UMAX.

d.

given x & 15 == 11,

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

False

- The final comparison against 0 effectively checks if the most significant bit of the left hand sign is 0 or not.
- By the given statement, we know that the 4 least significant bits (lsb) of x are 1011. Thus (x >> 3) has a lsb of 1, while (x >> 2) has a lsb of 0.
- AND-ing the two together has a lsb of 0, which when negated is 1.
- Left-shifting by 31 thus results in a number with a most significant bit of 1, and the remaining bits being 0

- This is a negative number

e.

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

x + ux < 0

False

- In an addition of an unsigned integer with a signed integer, the signed integer is implicitly cast to unsigned.
- Thus, the addition of two unsigned integers will always be non-negative
    - This is regardless of the given

f.

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

((x | y) >> 30) == -1

False

- Per the given, we know that the two most significant bits of x and y can be either 10 and 10, 11 and 10, or 10 and 11.
- In the case where x and y are 10 and 10, (x | y) would have most significant bits of 10
- In that case, Right shifting (x | y) by 30 would the result in -2

2.

Given: x has a 4 byte value of 255

What is the value of the byte with the lowest address in a

255 is represented as 0x000000FF

a.

big endian system?

0x00

b.

little endian system?

0xFF