CSE351
Section 2 Quick Check

Name: _____

1) Pointers Check

Below x is being stored in little endian format as x's hex encoding is

351 = 0x15F = 0x000000000000015F

Hence we expect the pointer p to store the address of x, namely 0xB1E8 in the same format

Define:

```
long  x  = 351;
long *p  = &x;
```

Let's say x is stored at address 0xB1E8, and p is stored at address 0xB0.

0xB1E8	0xB1E9	0xB1EA	0xB1EB	0xB1EC	0xB1ED	0xB1EE	0xB1EF
5F	01	00	00	00	00	00	00

0xB0	0xB1	0xB2	0xB3	0xB4	0xB5	0xB6	0xB7
E8	B1	00	00	00	00	00	00

a) Fill in the 8 blank bytes of memory at address 0xB0
b) Fill in the following values in hex:

&p: 0xB0 *p: x = 0x15F

c) Declare:

```
long y = * (long *) x;
```

Do we know what the value of y is? (circle) Yes No

2) Endianness Check

The above line says to treat the value x = 0xB1E8 like its referring to an address in memory, and assign y to the value in that spot in memory. But we have no idea what data is at address 0xB1E8 in memory

Consider the following 8 bytes of memory from an x86-64 machine:

0xF0	0xF1	0xF2	0xF3	0xF4	0xF5	0xF6	0xF7
A0	B1	C2	D3	E4	F5	B1	E9

Declare:

```
short *p = 0xF2;
int *q = 0xF4;
long *r = 0xF0;
```

Fill in the blanks below in big endian. Remember memory in an x86-64 machine is little endian!

*p: 0xD3C2 *q: 0xE9B1F5E4 *r: 0xE9B1F5E4D3C2B1A0