- **Extra credit quiz**

```
const   long MAX_LEN = 20000;
char    buf[MAX_LEN];

if (len < MAX_LEN) {
        strcpy(buf, input);
}
```

Can a buffer overflow attack occur? If so, how long does input need to be?

To cause a buffer overflow, the length of input needs to be larger than the maximum value that can be stored in a short int (ranger of -32768 to 32767); therefore, the length needs to be at least 32768. However, not any value that overflows the short int will work, as the resulting value also needs to be less than 20000 (MAX_LEN) when converted to a short int.

Lower bound: 32768 (one more than the maximum value for a short int)

strlen of 32768 to short int

32768 in binary -> 1000 0000 0000 0000

Since the first digit is a 1, the resulting number will be negative.

2's compliment -> 0111 1111 1111 1111
+1
->1000 0000 0000 0000

This translates to 32768 and with the original first byte negative, a short int value of -32768

-32768 is less than MAX_LEN, so this will overflow the buffer.

Upper bound: 85535 (19999 plus the range of the value for a short int 65536)

strlen of 85535 to short int:

85535 in binary -> 0001 0100 1110 0001 1111

Since it is too large to fit in a short int, reject the high byte

-> 0100 1100 0001 1111

This translates to 19999. Since this is less than MAX_LEN (20000), the buffer will overflow.

Hence, the first range of values for the length of input that will cause a buffer overflow are 32768 to 85535. Any positive multiple of 65536 (the range of the value for a short int) added to those ranges will also cause a buffer overflow. For example, a number in the range of 98304 (32,768 + 65536) to 151071 (85535 + 65536) or 229376 (32768 + 3 * 65536) to 282143 (85535 + 3 * 65536) will also cause a buffer overflow.