QUESTION #1

Suppose that you wrote the following statements in C/C++;

```c
int my_int_01;
unsigned int my_int_02;

void main (void)
{
    my_int_02 = 63489;
    if (my_int_02 <= 32767)
    {
        my_int_01 = my_int_02;
    }
    else
    {
        warning …
    }

    printf("my integer: %d\n", my_int_01);
}
```

Assuming that the computers that run this program is a 16-bit architecture computer (i.e., the registers in the processors are all 16 bits), what number will you see when you print the value of “my_int_01” in the program? Show all your work to find the number (i.e., please explain how you got the number). Note that “%d” in the `printf` statement means “print it as a decimal number”.

my_int_02 = 63489\textsubscript{(10)} = \textblacksquare{1111 1000 0000 0001}_2

my_int_01 = \textblacksquare{1111 1000 0000 0001}_2 \odot

**the largest positive \# 2’s complement integer can represent:**

0111 1111 1111 1111 = 32,767

Binary → Decimal for 2’s complement integers

\begin{align*}
1111100000000001_2 &= 1000000000000000_2 + 011110000000000001_2 \\
&= -32,768 + 30721 \\
&= -2,047
\end{align*}

**Conclusion:** -2,047 \odot
QUESTION #2

For 16-bit architecture computers, what is the largest positive integer the processors can handle? What is the smallest negative integer the processors can handle?

**Unsigned integers:** 0 through \((2^{16} - 1) = 0 \text{ through } 65,535\)

\(2^{16}\) different binary bit patterns:

4 bits: MSB \(2^3\) \(2^2\) \(2^1\) \(2^0\) LSB \(-2^3, 2^2, 2^1, \ldots, 2^0\)

\[1 \ 1 \ 1 \ , \ 1 = 2^{15} + 2^{14} + 2^{13} \ldots + 2^0 = 65,535\]

**Two's complement integers:** \(-2^{15}\) through \((2^{15} - 1) = -32,768 \text{ through } 32,767\)
QUESTION #3

Transform the following decimal number to the two’s complement binary number (using the 16-bit format: your processor is a 16-bit architecture processor): \(-63_{(10)}\). Show all your work.

**Step 1:** Find the binary bit pattern for its positive counterpart:

\[63_{(10)} = 0000000000111111_{(2)}\]

**Step 2:** Invert the bit pattern

\[
\begin{align*}
0000000000111111_{(2)} \\
1111111111000000_{(2)}
\end{align*}
\]

**Step 3:** Add “1” at the LSB

\[
\begin{align*}
1111111111000000_{(2)} \\
0000000000000001
\end{align*}
\]

\[111111111100001\]
QUESTION #4

Transform the following decimal number to the two’s complement binary number (using the 32-bit format: your processor is a 32-bit architecture processor): \(-95\)\(_{10}\). Show all your work.

QUESTION #5

How the binary bit pattern of \(-63\)\(_{10}\) for a 16-bit processor can be extended for a 32-bit register?

\[111111111100001\] = \[11111111111111111100001\] -> -63 in 32 bit
\[0000000011111\] = \[00000000000000000000000011111\] -> 63 in 32-bit