a. (2 pts) Convert -28 to an 8-bit two’s complement number.

\[ +28 = 0x \ 1C \quad (1 \times 16 + 12 = 28) \]
\[ -28 = 0 - (+28) \]
\[ = 0x00 - 0x1C = 0xE4 \]

b. (2 pts) The value \( 0xE9 \) is an 8-bit two’s complement number, give its decimal value.

**MSbit of 0xE9 is ‘1’, so this is a negative number, so we know the sign.**

To get magnitude, we know that \( +N = 0 - (-N) \), so

\[ +N = 0x00 - 0xE9 = 0x17 = +23 \]

so this number \( 0xE9 \) is \(-23\)!

c. (3 pts) Write PIC18 assembly code to implement the following C code fragment.

**Please note that k is an INT!!!**

\[
\begin{align*}
\text{int } k; \\
bcf & \quad \text{STATUS, C} \quad ; \text{shift in a ‘0’} \\
rclf & \quad k,f \quad ; \text{left shift LSByte first} \\
k = k << 1; \\
rclf & \quad k+1,f \quad ; \text{left shift MSByte next}
\end{align*}
\]

d. (3 pts) Fill in the blanks below in converting the C code to assembly language.

**Please note that j, k are INT !!!!!**

\[
\begin{align*}
\text{int } j, k; \\
d\left(\begin{array}{c}
\text{operation 1...} \\
\text{operation 2...} \\
\text{while}(k \neq j)
\end{array}\right)
\end{align*}
\]

**loop_top:***

....operation 1...
.... operation 1....

; subtract each byte
movf \ j, w
subwf \ k, w ; LSByte k – LSByte j
bnz \ loop_top ; if not zero, entire 16-bit value not equal
movf \ j+1, w
subwf \ k+1, w ; MSByte k – MSByte j
bnz \ loop_otp ; if not zero, entire 16-bit value not equal