You may NOT use a calculator. Assume the following memory/register contents at the beginning of each instruction:

<table>
<thead>
<tr>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x040</td>
<td>0x7A</td>
</tr>
<tr>
<td>0x041</td>
<td>0x4B</td>
</tr>
<tr>
<td>0x042</td>
<td>0xFC</td>
</tr>
<tr>
<td>0x043</td>
<td>0x1D</td>
</tr>
</tbody>
</table>

a. (3 pts) In the code below, give the FINAL values of FSR0, FSR1 and the final value of any changed memory locations after the instruction sequence is executed.

```
lsr       FSR0, 0x041
lsr       FSR1, 0x043
movff     PREINC0, POSTDEC1
```

**PREINC0** means to pre-increment FSR0, then access the contents of the memory location pointed to by FSR0. So, incrementing FSR0++ produces 0x042. **POSTDEC1** means to access the contents of the mem. location pointed to by FSR1, then decrement FSR1. So the `movff` instruction does the following move:

```
movff     0x042, 0x043     ; so location [0x042] → 0x043, or 0xFC → 0x43
```

After the `movff`, the value of FSR1 is decremented (**POSTDEC1**), so FSR1-- = 0x42.

The final results are:

mem location 0x043 changed to 0xFC. Final value of FSR1 = 0x042. Final value of FSR0 = 0x042

b. (3 pts) In the code below, give the FINAL values of FSR0 and the final value of any changed memory locations after the instruction sequence is executed.

```
lsr       FSR0, 0x40
movlw     3
movff     PLUSW0, INDF0
```

**PLUSW0** means to access the contents of the memory location pointed to by [FSR0 + W].

**INDF0** means to access the contents of the mem. location pointed to by FSR0. So the `movff` instruction does the following move:

```
movff     [0x040+W], 0x040     ; so location [0x040 + 3] → 0x040, or [0x043] → 0x040, or 0x1D → 0x040
```

The PLUSW0 mode **does NOT** change the value of FSR0.

The final results are:

mem location 0x043 changed to 0x1D. Final value of FSR0 = 0x040. Final value of W is 3.
c. (2 pts) Write the C function below in PIC18 assembly language. Assume that the parameters for \( ptr0, ptr1 \) are PASSED into the function within the FSR0, FSR1 registers respectively. What this code does is copy the value of the \textbf{int} pointed to by the pointer \( ptr0 \) to the location pointed to by \( ptr1 \). DO NOT FORGET THAT THIS IS A SUBROUTINE! Also, do not forget that these are pointers to INT values!

\begin{verbatim}
mysub (int *ptr0, int *ptr1) {
    *ptr1 = *ptr0;
}
\end{verbatim}

A solution:

\begin{verbatim}
movff POSTINC0, POSTINC1 ; copy LSByte of integer value. Use POSTINC to point at next byte
movff INDF0, INDF1       ;copy MSByte of integer value
return                   ; return from subroutine
\end{verbatim}

WRONG ANSWER: (this just copies FSR0 to FSR1. This implements the C code: \( ptr1 = ptr0; \))

\begin{verbatim}
movff FSR0L, FSR1L
movff FSR0H, FSR1H
return
\end{verbatim}

WRONG ANSWER: (no such thing as INDF0L, INDF0H).

\begin{verbatim}
movff INDF0L, INDF0L
movff INDF0H, INDF0H
return
\end{verbatim}