1) The following MAL code pops off the system stack into $s0. It is unsafe under some circumstances. Give one instance of where it would fail, and rewrite the code so that it is safe. (3 points)
   add $sp, $sp, 4
   lw $s0, ($sp)

This will fail if an interrupt occurs between the add and lw. We can make this safe by extracting the value from the stack first, THEN changing the stack pointer.

2) Perform the following computations, assuming an 8-bit register using the indicated representation. Indicate overflow (if overflow, you STILL need to add!). (3 points)

<table>
<thead>
<tr>
<th>2's Complement</th>
<th>Unsigned</th>
<th>2's Complement</th>
<th>Unsigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101 1110</td>
<td>1100 1010</td>
<td>0x5a</td>
<td>0x3c</td>
</tr>
<tr>
<td>+ 1010 1010</td>
<td>+ 0011 1001</td>
<td>+ 0x42</td>
<td>+ 0xa2</td>
</tr>
<tr>
<td>1000 1000</td>
<td>OF 10000 0011</td>
<td>OF 0x9c</td>
<td>0xde</td>
</tr>
</tbody>
</table>

3) Step through the following code and familiarize yourself with its function. Describe what the code is intended to do (hint: you can say what it's doing in 1 sentence). Now describe why it will not always behave in the intended way. (2 points + 2 points)

The code is meant to take lower-case characters and convert them into upper-case, or take upper-case characters and convert them into lower-case. It runs into problems when $s0 contains a non-alpha char that is either <'A' or '>'z'. In either of these cases, the conversion still occurs even though the char is not an alpha.

getc $s0
while:
   bne $s0, '\n', endwhile
jal mysteryfunc
putc $s0
getc $s0
j while
endwhile:
   .
   .

mysteryfunc:
   bge $s0, 'a', lower
   ble $s0, 'Z', upper
   jr $ra
lower:
   sub $s0, $s0, 'a'
   add $s0, $s0, 'A'
   jr $ra
upper:
   sub $s0, $s0, 'A'
   add $s0, $s0, 'a'
   jr $ra