QUESTION ONE:
A 32 bit word can represent a MIPS instruction. How can we tell if that instruction is of type R, J, or I?

QUESTION TWO:
Below is a pretty neat function (subroutine) written in MIPS that’s *almost* ready for prime time – but eight key lines of code are missing. Get busy!

```
sum_tree:
____________________
____________________
____________________
____________________
add $v0, $0, $0
beq $a0, $0, done
lw $s0, 0($a0)
add $s1, $a0, $0
lw $a0, 4($a0)
jal sum_tree
add $s0, $s0, $v0
lw $a0, 8($s1)
jal sum_tree
add $s0, $s0, $v0
done:
____________________
____________________
____________________
____________________
```

What’s this code up to, anyway?

QUESTION THREE:
Assuming the first instruction of each snippet is located at memory address 0x10001000, fill in this table.

<table>
<thead>
<tr>
<th>MIPS Snippet</th>
<th>Instruction as binary</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>addu $t0, $t0, $t0</code></td>
<td>0b 0000 0000 0000 0000 0000 0000 0000 0000</td>
</tr>
<tr>
<td><code>Self: j Self</code></td>
<td>0b 0100 0001 0000 0100 0000 0000 0000 0000</td>
</tr>
<tr>
<td><code>bne $t0, $t1, Skip</code></td>
<td>0b 0100 0001 0000 0100 0000 0000 0000 0000</td>
</tr>
<tr>
<td><code>addiu $v0, $zero, 1</code></td>
<td>0b 0100 0001 0000 0100 0000 0000 0000 0000</td>
</tr>
<tr>
<td><code>Skip:</code></td>
<td>0b 0100 0001 0000 0100 0000 0000 0000 0000</td>
</tr>
<tr>
<td><code>li $t0, 70000</code></td>
<td>0b 0000 1101 0000 0000 0000 0000 0000 0000</td>
</tr>
</tbody>
</table>