1) (15 pts) Write a MAL code procedure that calculates the factorial of the number passed in $a0. The return value should be in $v0. Use recursion and use the system stack to store the return address. For example, if 5 is in $a0 then return 5x4x3x2x1= 120 in $v0.

```mal
# assumes that a value >= 1 is passed in $a0
factorial:  bne  $a0, 1, continue
            li   $v0, 1
            jr   $ra
continue:  sub  $sp, $sp, 8  # make room on stack
            sw   $ra, 4($sp)  # store return address
            sw   $a0, 8($sp)  # store value
            sub  $a0, $a0, 1
            jal  factorial
            lw   $t0, 8($sp)  # get number off stack
            mul  $v0, $t0, $v0
            lw   $ra, 4($sp)  # get return address
            jr   $ra
```

2) (10 pts) Hand assemble the following MAL code. Start instructions at address 0x0004 4800.

```mal
and   $4, $5, $8
beq   $4, $0, bob
lui   $20, 0x66aa
bob:   lb   $9, -8($20)
```

Or in Hex

```
0x00A8 2024 (and)
0x1080 0004 (beq)
0x3C14 66AA (lui)
0x8289 FFF8 (lb)
```
3) (15 pts) Hand assemble the following MAL .text section. Start instructions at address 0x0008 8800 and data starts at 0x0004 4400. Show what the .data section would also look like.

```
.value: .word 12

.loop: getc $8      # convert to TAL
       syscall
       la $9, value
       bltz $9, $0, loop
       done           # convert to TAL
       syscall
```

4) (5 pts) Disassemble the following MIPS RISC code. Make up label names when needed. The code starts at address 0x0040 0000.

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>TAL Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0040 0000</td>
<td>0x41E6 0000</td>
<td>Illegal instruction</td>
</tr>
<tr>
<td>0x0040 0004</td>
<td>0x9287 1020</td>
<td>lbu $7, 4128($20)</td>
</tr>
<tr>
<td>0x0040 0008</td>
<td>0x00AB A806</td>
<td>srlv $21, $11, $5</td>
</tr>
<tr>
<td>0x0040 000C</td>
<td>0x0681 ffffd</td>
<td>bgez $20, -3</td>
</tr>
</tbody>
</table>

5) (5 pts) Give a correct TAL translation of the virtual instruction

```}

```
la $15, variable

if "variable" has been assigned the address 0x0024 6088

    lui $15, 0x0024
    ori $15, 0x6088
```
6) (5 pts) How can the MAL “move” instruction be implemented in TAL?

“add”, “or”, “and”, “sub”, etc instructions with $0$ register

7) (5 pts) Give two ways that the MAL instruction

\[
\text{blt} \quad $3, $18, \text{branch}
\]

can be translated into TAL instructions.

\[
\text{sub} \quad $7, $3, $18 \\
\text{bltz} \quad $7, \text{branch}
\]
\text{OR}

\[
\text{sub} \quad $7, $18, $3 \\
\text{bgez} \quad $7, \text{branch}
\]

8) (5 pts) What can an assembler do if a calculated branch offset is too large to fit into the offset field of an instruction?

The assembly could use a “j” instruction which would give it a longer jump range ($2^{26}$ words displacement verses $2^{16}$ words displacement). If that is not enough then the assembler could be tricky and load a 32bit address into a register and do a “jr” to that register, giving it the ability to go to any address.