Question 1) Translate the following to MIPS code to C. You can use either bne (branch on not equal) or beq (branch on equal)

```
if ( s0 == s1 )
        s0 = s0 + 1;
else
        s1 = s1 - 1;
s1 = s1 + s0;
```

```
bne $s0, $s1, ELSE
addi $s0, $s0, 1
j AFTER
ELSE: addi $s1, $s1, -1
AFTER: add $s1, $s1, $s0
```

Question 2) Translate the following code to MIPS code to C. This time you have to use beq (branch on equal)

```
if ( s0 == s1 )
        s0 = s0 + 1;
s1 = s1 - 1;
```

```
beq $s0, $s1, IF
j AFTER
IF: addi $s0, $s0, 1
AFTER: addi $s1, $s1, -1
```

Question 3) Convert the following C to MIPS:

```
while (t2 < t6)
{
        t5 = s4 + 12;
        t2 = t2 + 2;
}
```

```
LOOP: slt $t0, $t2, $t6
beq $t0, $zero, AFTER
addi $t5, $s4, 12
addi $t2, $t2, 2
j LOOP
AFTER:
```

Question 4) Convert the following C to MIPS:

```
if (t2 >= s4)
{
        t5 = s1 + 9;
        t6 = s1 - s7;
}
else
        t6 = t6 + 17;
```

```
slt $t0, $t2, $s4
bne $t0, $zero, ELSE
addi $t5, $s1, 9
sub $t6, $s1, $s7
j AFTER
ELSE: addi $t6, $t6, 17
AFTER:
```

Question 5) Assuming that the size of a short int is 2 bytes. Convert the following C to MIPS:

```
void main()
{
        short int my_array[3];
        my_array[0] = 6;
        my_array[1] = 16;
        my_array[2] = 7;
}
```

```
.text
        main:
                addi $sp, $sp, -6

                addi $t0, $zero, 6
                sh $t0, 0($sp)

                addi $t0, $zero, 16
                sh $t0, 2($sp)

                addi $t0, $zero, 7
                sh $t0, 4($sp)
```