MIPS cheat sheet

Instruction	Syntax	Example
add	add dest, src0, src1	add $s0, $s1, $s2
sub	sub dest, src0, src1	sub $s0, $s1, $s2
addi	addi dest, src0, immediate	addi $s0, $s1, 12
lw	lw dest, offset(base addr)	lw $t0, 4($s0)
sw	sw src, offset(base addr)	sw $t0, 4($s0)
bne	bne src0, src1, branchAddr	bne $t0, $t1, notEq
beq	beq src0, src1, branchAddr	bne $t0, $t1, Eq
j	j jumpAddr	j jumpWhenDone

C	MIPS

`// $s0 -> a, $s1 -> b`
`// $s2 -> c, $s3 -> z`
 	
`int a=4, b=5, c=6, z;`
`z = a+b+c+10;`
	`addi $s0, $0, 4`
	`addi $s1, $0, 5`
	`addi $s2, $0, 6`
	`add $s3, $s0, $s1`
	`add $s3, $s3, $s2`
	`addi $s3, $s3, 10`

// $s0 -> int *p = (int *)malloc
// (3*sizeof(int));
// $s1 -> a
`p[0] = 0;`
`int a = 2;`
`p[1] = a;`
`p[a] = a;`
	`sw $0, 0($s0)`
	`addiu $s1, $0, 2`
	`sw $s1, 4($s0)`
	`sll $t0, $s1, 2 #same as <<`
	`addu $t1, $t0, $s0`
	`sw $s1, 0($t1)`

`// $s0 -> a, $s1 -> b`
`int a = 5, b = 10;`
`if (a + a == b) {`
`  a = 0;`
`} else {`
`  b = a - 1;`
`}`
	`addiu $s0, $0, 5`
	`addiu $s1, $0, 10`
	`add $t0, $s0, $s0`
	`bne $t0, $s1, else`
	`add $s0, $0, $0`
	`j exit`
	`else: addiu $s1, $s0, -1`
	`exit: # done!`

`/*What does this do?`
`  (Not C, in English) */`
Returns 2^{30}, or 2^N where N is the immediate on line 3
Or crash for ancient system with less than 32 bit registers (in this case, how to fix the crash?)
	`addi $s0, $0, 0`
	`addi $s1, $0, 1`
	`addi $t0, $0, 30`
	`loop: beq $s0, $t0, done`
	`add $s1, $s1, $s1`
	`addi $s0, $s0, 1`
	`j loop`
	`done: # done!`

```
// Nth_Fibonacci(N):
// $s0 -> N, $s1 -> fib
// $t0 -> i, $t1 -> j
if(N==0) return 0;
else if(N==1) return 1;
N-=2;
int fib=1, i=1, j=1;
while(N!=0){
    fib = i+j;
    j = i;
    i = fib;
    N--;
}
return fib;
```

```
zero: bne  $s0, $0,  one
      addi $s1, $0,  0
      j    done
one:  addi $s1, $0,  1
      bne  $s0, $s1, init
      j    done
init: subi $s0, $s0, 2
      addi $t0, $0,  1
      addi $t1, $0,  1
loop: beq  $s0, $0,  done
      add  $s1, $t0, $t1
      addi $t1, $t0, 0
      addi $t0, $s1, 0
      subi $s0, $s0, 1
      j    loop
done: # done!
```

```
int a[10];
int i;
int total = 0;
for (i = 0; i < 10; i+=2) {
    temp = a[i];
    a[i] = a[i+1];
    total = total + a[i] + a[i+1];
    a[i+1] = total;
}
```

```
      add  $s0, $0,  $0
      addi $s1, $0,  10
      add  $s2, $0,  $0
loop: beq  $s0, $s1, done
      addi $t0, $s0, 4
      lw   $t0, 0($s0)
      lw   $t1, 4($s0)
      sw   $t1, 0($s0)
      add  $s2, $s2, $t0
      add  $s2, $s2, $t1
      sw   $s2, 4($s0)
      addi $s0, $s0, 1
      j    add
add:  addi $s0, $s0, 1
      j    loop
done: # done!
```

Fill in the blanks in the MIPS code. Also add jump labels in appropriate places

```
// 0x100 -> &a, 0x200 -> &b
// $s0 -> i

int a[4], b[4];
int i;
for (i = 4; i != 0; i--) {
    b[i] = a[i];
}
```

```
          addi $s0, $0,  4
loop:     beq  $s0, $0,  done
          add  $t0, $0,  4
          add  $t1, $0,  $s0
do_mult:  beq  $t0, $0,  copy
          add  $t1, $t1, $t1
          sub  $t0, $t0, 1
          j    do_mult
copy:     lw   $t0, 0x100($t1)
          sw   $t0, 0x200($t1)
          subi $s0, $s0, 1
          j    loop
done:     # We are done
```

Bonus: There's a building with 100 floors. You have 2 eggs. Assume that the eggs are of the same attributes. At the worst, how many times do you have to drop an egg off the building (count drops of both eggs) in order to determine the lowest floor at which the eggs will break? (Hint: it's not 19)

14. Look online or ask me. Don't feel like typing it out, lol (Basically a decimal-nary search kind of thingy)