1. **Status Flags**

Suppose your registers have the following values:

- r0 = `0xffffffff`
- r1 = `0x00000001`
- r2 = `0x00000003`
- r3 = `0xfffffff0`

The N, Z, C, and V flags start out as zero.

After the execution of the following instructions, fill in the values that the Negative, Zero, Carry, and oVerflow flags would have. (Assume each of these runs individually, i.e. start over from scratch each time, these are not part of a program).

(a) `ADD r4, r0, r2, ASR #3`

N	Z	C	V

(b) `ADDS r4, r0, r1`

N	Z	C	V

(c) `LSRS r4, r0, #1`

N	Z	C	V

(d) `ANDS r4, r0, r3`

N	Z	C	V

(e) `CMP r2, #3`

N	Z	C	V

2. Status Flags (Solution)

Suppose your registers have the following values:

- r0 = `0xffffffff`
- r1 = `0x00000001`
- r2 = `0x00000003`
- r3 = `0xfffffff0`

The N, Z, C, and V flags start out as zero.

After the execution of the following instructions, fill in the values that the Negative, Zero, Carry, and oVerflow flags would have. (Assume each of these runs individually, i.e. start over from scratch each time, these are not part of a program).

(a) `ADD r4, r0, r2, ASR #3`

N	Z	C	V
0	0	0	0

plain ADD doesn't update flags, need ADDS for that

(b) `ADDS r4, r0, r1`

N	Z	C	V
0	1	1	0

r4=r0+r1=0xffffffff+0x00000001 = 0x00000000 overflow to C
both carry into high bit=1 and carry out of high bit=1 so V=0

(c) `LSRS r4, r0, #1`

N	Z	C	V
0	0	1	0

r4=r0 logical shift right by 1, r4=0x7fffffff, low bit into C=1

(d) `ANDS r4, r0, r3`

N	Z	C	V
1	0	0	0

r4=r0 bitwise and r3, r0=0xffffffff & 0xfffffff0 = 0xfffffff0
since the high bit of the result is set, it's considered negative

(e) `CMP r2, #3`

N	Z	C	V
0	1	1	0

compare r2 with 3. r2 is 3. Same as doing a subtract. This is tricky, do the subtract as twos complement add and you'll find that 3+-3 0x00000003+0xfffffffd= 0 but also overflows carry 3-3=0