4.

(a) if(x < y && y < z) z = 6; else z = x;

```
        LDR   R0, x
        LDR   R1, y
        LDR   R2, z
        CMP   R0,R1
        BGE   Else          // R0 >= R1, x >= y
        CMP   R1, R2
        BGE   Else          // R1 >= R2, y >= z
Then:   LDR   R2, =6        // z = 6
        STR   R2, z
        B     EndIf
Else:   LDR   R2, x         // z = x
        STR   R2, z
EndIf:  BX    LR
```

(b) if(-10 < x && x < +10) goto L1;

```
        LDR   R0, x
        CMP   R0, -10
        BLE   EndIf         // R0 <= -10, x <= -10
        CMP   R0, 10
        BGE   EndIf         // R0 >= +10, x >= +10
Then:   B     L1            // goto L1
```

(c) if(x < 10 || x > 20) y = 0; else y = 1;

```
        LDR   R0, x
        CMP   R0, 10
        BLT   Then          // R0 < 10, x < 10
        CMP   R0, 20
        BGT   Then          // R0 > 20, x > 20
Then:   LDR   R1, =0
        STR   R1, y
        B     EndIf
Else:   LDR   R1, =1
        STR   R1, y
Endif:  BX    LR
```

(d) if('a' <= ch && ch <= 'z')
        ch = ch - 'a' + 'A';

```
        LDR   R0, CH
        CMP   R0, ‘a’
        BLT   EndIf          // R0 < ‘a’, ch < ‘a’
        CMP   R0, ‘z’
        BGT   EndIf          // R0 > ‘z’, ch > ‘z’
Then:   ADD   R0, R0, ‘A’    // R0 = R0 + ‘A’
        SUB   R0, R0, ‘a’    // R0 = R0 + ‘A’ - ‘a’
        STR   R0, ch
EndIf:  BX    LR
```

(e) x = y / 5;

```
        LDR   R0, y
        LDR   R1, =5
        DIV   R0, R0, R1     //signed or unsigned?
        STR   R0, x
```

(f)
```
uint32_t u32;
int32_t s32;
if(u32 > 10) s32 = s32 - 1;
else s32 = s32 + 1;
```

```
        LDR   R0, u32
        LDR   R1, s32
        CMP   R0, 10
        BLE   Else           // u32 <= 10
Then:   SUB   R1, R1, 1
        STR   R1, s32
        B     EndIf
Else:   ADD   R1, R1, 1
        STR   R1, s32
        B     EndIf
EndIf:  BX    LR
```

I want to try IT block as well

```
        LDR     R0, u32
        LDR     R1, s32
        CMP     R0, 10
        ITE     GT
        SUBGT   R1, R1, 1     // if (R0 > 10)
        ADDLE   R1, R1, 1     // if (R0 <= 10)
        STR     R1, s32
        BX      LR
```

(g) int32_t s32;
if(-10 < s32 && s32 < +10) s32 = 0;

```
        LDR   R0, s32
        CMP   R0, -10
        BLE   EndIf          // if (R0 <= -10), s32 <= -10
        CMP   R0, +10
        BGE   EndIf          // if (R0 >= +10), s32 >= +10
Then:   LDR   R0, = 0
        STR   R0, s32
        B     EndIf
EndIf:  BX    LR
```

(h) uint32_t u32, min, max;
if(u32 < min || u32 > max) u32 = 0;

```
        LDR   R0, u32
        LDR   R1, min
        LDR   R2, max
        CMP   R0, R1
        BLT   Then           // if (R0 < R1), u32 < min
        CMP   R0, R2
        BGT   Then           // if ( R0 > R2), u32 > max
Then:   LDR   R0, = 0
        STR   R0, u32
        BX    LR
```

5.
int32_t Minimum(int32_t data[ ], int32_t count);
// what is count? I'll just assume it is the size of the array

```
Minimum:    //R0 = data[ ], R1 = count
            //my idea is to load data[0] into R0, load data[1] into R1, compare R0 and R1,
keep the smaller on in R0, and then load data[2] into R1, and compare
```

```
    initialization ;
L1: if (!condition) goto L2 ;
    loopBody ;
    update ;
    goto L1 ;    // repeat
L2:
```

```
L0:  if (condition1) goto L1 ;
     if (condition2) goto L1 ;
     goto L2 ;
L1:  loopBody ;
     goto L0 ;
L2:
```

```
        ADR   R0, R0                // R0 = data[0]
        LDR   R1, [R0, R0, LSL 2]   // R1 = R0 + 4R0 = data[1]
L0:     //how to set the boundary?
        CMP   R0, R1
        BLE   L1                    // if (R0 <= R1), data[0] <= data[1]
L1:     // keep R0, and load data[2] into R1
        LDR   R1, [R1, R0, LSL 2]   // R1 = data[2]
        B     L0                    // goto L0 to compare
L2:     // keep R1, and load data[2] into R0
        LDR   R0, [R1, R0, LSL 2]   // R0 = data[2]
        B     L0
```