1. Translate each of the following C statements into a corresponding sequence of ARM Cortex-M4 instructions, where *x*, *y*, and *z* are variables of type int32_t:

(a) `z = (x < y) ? 6 : x ;`

```
        LDR     R0,x            // x < y ?
        LDR     R1,y
        CMP     R0,R1
        BGE     Else
Then:   LDR     R0,=6           // z = 6 ;
        STR     R0,z
        B       EndIf
Else:   LDR     R0,x            // z = x ;
        STR     R0,z
EndIf:
```

(b)
```
x = 0;
   for (y = 1; y < 1000; y = 2*y) x += y ;
```

```
        LDR     R0,=0       // x = 0;
        STR     R0,x
        LDR     R0,=1       // y = 1 ;
        STR     R0,y
   Top: LDR     R0,y        // y < 1000?
        CMP     R0,1000
        BGE     Done
        LDR     R1,x        // x += y ;
        ADD     R1,R1,R0
        STR     R1,x
        LSL     R0,R0,1     // y = 2 * y ;
        STR     R0,y
        B       Top         // repeat
   Done:
```

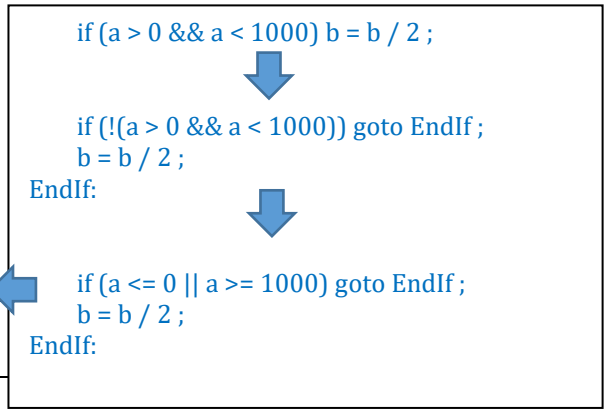
(c) `if (x > 10) { if (x < 20) y = 1 ; else z = 0 ; }`

```
        LDR     R0,x
        CMP     R0,10
        BLE     Done
        CMP     R0,20
        BGE     Else
Then:   LDR     R0,=1
        STR     R0,y
        B       Done
Else:   LDR     R0,=0
        STR     R0,z
Done:
```

2. Translate each of the following C statements into a corresponding sequence of ARM Cortex-M4 instructions without using IT blocks:

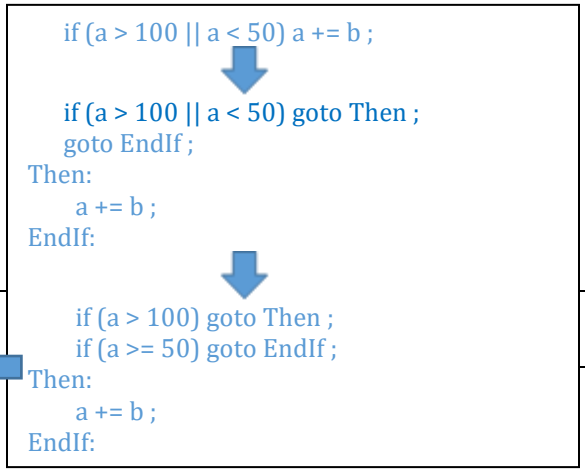
(a) uint16_t a, b ;

```
    if (a > 0 && a < 100) b = b / 2 ;

        LDRH    R0,a            // a > 0 ?
        CMP     R0,0
        BLS     EndIf
        CMP     R0,100          // a < 100 ?
        BHS     EndIf
        LDRH    R0,b            // b = b / 2 ;
        LDR     R1,=2
        UDIV    R0,R0,R1
        STRH    R0,b
EndIf:
```



(b) int32_t a, b ;

```
    if (a > 100 || a < 50) a += b ;

        LDR     R0,a            // a > 100 ?
        CMP     R0,100
        BGT     Then
        CMP     R0,50           // a < 50 ?
        BGE     EndIf
Then:   LDR     R0,a            // a += b ;
        LDR     R1,b
        ADD     R0,R0,R1
        STR     R0,a
EndIf:
```



3. Use an IT block to convert each of the following into a sequence of ARM Cortex-M4 Instructions:

(a) int64_t a, b, c ;

```
   if (a > b) c = b + 2 ;

        LDRD    R0,R1,a         // a > b ?
        LDRD    R2,R3,b
        SUBS    R0,R0,R2        // don't need the difference,
        SBCS    R1,R1,R3        // only need its characteristics in flags
        ITTTT   GT
        LDRDGT  R2,R3,b         // c = b + 2
        ADDSGT  R2,R2,2
        ADCGT   R3,R3,0
        STRDGT  R2,R3,c
EndIf:
```

(b)
```
uint64_t a, b, c ;

if (a == b) c = 0 ; else c = a - b ;
```

```
        LDRD    R0,R1,a       // a == b ?
        LDRD    R2,R3,b
        SUBS    R0,R0,R2      // keep the difference for later
        SBCS    R1,R1,R3      // need its characteristics in flags now
        ITTE    EQ
        LDREQ   R0,=0         // c = 0 ;
        STRDEQ  R0,R0,c
        STRDNE  R0,R1,c       // store the difference (a – b) in c
```

(c)
```
uint64_t a, b, c ;

a = (b < c) ? b : c ;
```

```
        LDRD    R0,R1,b       // b < c ?
        LDRD    R2,R3,c
        SUBS    R4,R0,R2      // don't need the difference,
        SBCS    R4,R1,R3      // only need its characteristics in flags
        ITE     LO
        STRDLO  R0,R1,a       // a = b ;
        STRDHS  R2,R3,a       // a = c ;
```

4. Translate each of the following C statements into a corresponding sequence of ARM Cortex-M4 instructions, where *ch* is the label on an 8-bit memory location whose content is an ASCII character, and *x*, *y*, and *z* are labels on 32-bit variables of type int32_t:

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

```
        LDR     R0,x      // x < y ?
        LDR     R1,y
        CMP     R0,R1
        BGE     Else
        LDR     R0,z      // y < z ?
        CMP     R1,R0
        BGE     Else
Then:   LDR     R0,=6     // z = 6 ;
        STR     R0,z
        B       EndIf
Else:   LDR     R0,x      // z = x ;
        STR     R0,z
EndIf:
```

```
        if (x < y && y < z) goto Then ;
        goto Else ;
Then:   z = 6 ;
        goto EndIf ;
Else:   z = x ;
EndIf:

        if (!(x < y && y < z)) goto Else ;
Then:   z = 6 ;
        goto EndIf ;
Else:   z = x ;
EndIf:

        if (x >= y) goto Else ;
        if (y >= z) goto Else ;
Then:   z = 6 ;
        goto EndIf ;
Else:   z = x ;
EndIf:
```

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

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

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

```
        LDR   R0,x            // x < 10 ?
        CMP   R0,10
        BLT   Then
        CMP   R0,20           // x > 20 ?
        BLE   Else
Then:   LDR   R0,=0           // y = 0 ;
        STR   R0,y
        B     EndIf
Else:   LDR   R0,=1           // y = 1 ;
        STR   R0,y
EndIf:
```

```
        if (x < 10 || x > 20) goto Then ;
        goto Else ;
Then:   y = 0 ;
        goto EndIf ;
Else:   y = 1 ;
EndIf:
```

```
        if (x < 10) goto Then ;
        if (x <= 20) goto Else ;
Then:   y = 0 ;
        goto EndIf ;
Else:   y = 1 ;
EndIf:
```

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

```
        LDRB    R0,ch         // ch >= 'a'
        CMP     R0,'a'
        BLT     EndIf
        CMP     R0,'z'        // ch <= 'z' ?
        BGT     EndIf
Then:   SUB     R0,R0,'a'     // ch = ch-'a'+'A'
        ADD     R0,R0,'A'
        STRB    R0,ch
EndIf:
```

```
        if (!(ch >= 'a' && ch <= 'z') ) goto EndIf ;
        ch = ch – 'a' + 'A' ;
EndIf:
```

```
        if (ch < 'a' || ch > 'z' ) goto EndIf ;
        ch = ch – 'a' + 'A' ;
EndIf:
```

```
        if (ch < 'a' ) goto EndIf ;
        if (ch > 'z' ) goto EndIf ;
        ch = ch – 'a' + 'A' ;
EndIf:
```

(e) `x = y / 5 ;`

```
        LDR     R0,y
        LDR     R1,=5
        SDIV    R0,R0,R1
        STR     R0,x
```

(f)
```
uint32_t u32 ;
int32_t s32 ;

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

```
        LDR     R0,u32      // u32 > 10 ?
        CMP     R0,10
        BLS     Else
Then:   LDR     R0,s32      // s32 = s32 – 1 ;
        SUB     R0,R0,1
        STR     R0,s32
        B       EndIf
Else:   LDR     R0,s32      // s32 = s32 + 1 ;
        ADD     R0,R0,1
        STR     R0,s32
EndIf:
```

```
LDR     R0,s32
LDR     R1,u32
CMP     R1,10
ITE     HI
SUBHI   R0,R0,1
ADDLS   R0,R0,1
STR     R0,s32
```

(g)
```
int32_t s32 ;

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

```
        LDR     R0,s32          // s32 > -10
        CMP     R0,-10
        BLE     EndIf
        CMP     R0,10           // s32 < +10 ?
        BGE     EndIf
Then:   LDR     R0,=0           // s32 = 0 ;
        STR     R0,s32
EndIf:
```

```
        if (s32 > -10 && s32 < +10) goto Then;
        goto EndIf ;
Then:   s32 = 0 ;
EndIf:

        if (s32 <= -10 || s32 >= +10) goto EndIf;
Then:   s32 = 0 ;
EndIf:

        if (s32 <= -10)  goto EndIf;
        if (s32 >= +10) goto EndIf;
Then:   s32 = 0 ;
EndIf:
```

(h)
```
uint32_t u32, min, max ;

if (u32 < min || u32 > max) u32 = 0 ;
```

```
        LDR     R0,u32          // u32 < min ?
        LDR     R1,min
        CMP     R0,R1
        BLO     Then
        LDR     R1,max          // u32 > max ?
        CMP     R0,R1
        BLS     EndIf
Then:   LDR     R0,=0           // u32 = 0 ;
        STR     R0,u32
EndIf:
```

```
        if (u32 < min || u32 > max) goto Then ;
        goto EndIf ;
Then:   u32 = 0 ;
EndIf:

        if (u32 < min) goto Then ;
        if (u32 <= max) goto EndIf ;
Then:   u32 = 0 ;
EndIf:
```