CS1021 Tutorial 3

Q1 Translate each of the following pseudo-code statements into a sequence of ARM assembly language instructions. Assume x and y are signed integers and x is in R1 and y is in R2.

(i)

```
if (x== 0)
    x = x + 5;
```

```
        CMP   R1, #0        ; x == 0?
        BNE   L1            ; != (opposite condition to == in pseudo-code)
        ADD   R1, R1, #5    ; x = x + 5
L1      ...
```

(ii)

```
if (x >= 5)
    x = 0;
```

```
        CMP   R1, #5        ; x >= 5?
        BLT   L1            ; < (opposite condition to >= in pseudo-code)
        MOV   R1, #0        ; x = 0
L1      ...
```

(iii)

```
x = 10;
y = 5;
while (x > 0) {
    y = y*x;
    x = x - 1;
}
```

```
        MOV   R1, #10       ; x = 10
        MOV   R2, #5        ; y = 5
L1      CMP   R1, #0        ; x == 0?
        BLE   L2            ; <= (opposite condition to > in pseudo-code)
        MUL   R2, R1, R2    ; y = y*x
        SUB   R1, R1, #1    ; x = x – 1
        B     L1
L2      ...
```

(iv)

```
if (x < 9) {
    x = x + 1;
} else {
    x = 0;
}
```

```
        CMP   R1, #9        ; x < 9?
        BGE   L1            ; >= (opposite condition to < in pseudo-code)
        ADD   R1, R1, #1    ; x = x + 1
        B     L2            ; skip else
L1      MOV   R1, #0        ; x = 0
L2      …
```

(v)

```
if (x > 9) {
    x = 0;
    if (y > 9) {
        y = 0
    } else {
        y = y + 1;
    }
} else {
    x = x + 1;
}
```

```
        CMP   R1, #9        ; x > 9?
        BLE   L2            ; <= (opposite condition to > in pseudo-code)
        MOV   R1, #0        ; x = 0
        CMP   R2, #9        ; y > 9?
        BLE   L1            ; <= (opposite condition to > in pseudo-code)
        MOV   R2, #0        ; y = 0
        B     L3            ; skip else parts
L1      ADD   R2, R2, #1    ; y = y + 1
        B     L3            ; skip else part
L2      ADD   R1, R1, #1    ; x = x + 1
L3      …
```

Q2 Write an ARM assembly language program to compute x^y where x and y are unsigned integers. Assume x is in R1, y in R2 and the result is stored in R0.

```
        MOV R1, #2          ; test with x = 3
        MOV R2, #4          ; test with y =4
        MOV R0, #1          ; r = 1
L1      CMP  R2, #0         ; while (y != 0) ?
        BEQ  L2             ; == (opposite condition to != in pseudo-code)
        MUL  R0, R1, R0     ; r = r*x
        SUB  R2, R2, #1     ; y = y – 1
        B    L1             ; repeat
L2      …
```