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) \[\text{if } (x < 9) \{\]
\[
\hspace{1cm} x = x + 1;\]
\[
\} \text{ else } \{\]
\[
\hspace{1cm} x = 0;\]
\[
\}\]

\[
\begin{align*}
\text{CMP R1, #9} & \quad ; x < 9? \\
\text{BGE L1} & \quad ; \geq (\text{opposite condition to } < \text{ in pseudo-code}) \\
\text{ADD R1, R1, #1} & \quad ; x = x + 1 \\
\text{B L2} & \quad ; \text{skip else} \\
\text{L1 MOV R1, #0} & \quad ; x = 0 \\
\text{L2 ... } & \\
\end{align*}
\]

(v) \[\text{if } (x > 9) \{\]
\[
\hspace{1cm} x = 0;\]
\[
\text{if } (y > 9) \{\]
\[
\hspace{2cm} y = 0\]
\[
\} \text{ else } \{\]
\[
\hspace{2cm} y = y + 1;\]
\[
\} \text{ else } \{\]
\[
\hspace{2cm} x = x + 1;\]
\[
\}\]

\[
\begin{align*}
\text{CMP R1, #9} & \quad ; x > 9? \\
\text{BLE L2} & \quad ; \leq (\text{opposite condition to } > \text{ in pseudo-code}) \\
\text{MOV R1, #0} & \quad ; x = 0 \\
\text{CMP R2, #9} & \quad ; y > 9? \\
\text{BLE L1} & \quad ; \leq (\text{opposite condition to } > \text{ in pseudo-code}) \\
\text{MOV R2, #0} & \quad ; y = 0 \\
\text{B L3} & \quad ; \text{skip else parts} \\
\text{L1 ADD R2, R2, #1} & \quad ; y = y + 1 \\
\text{B L3} & \quad ; \text{skip else part} \\
\text{L2 ADD R1, R1, #1} & \quad ; x = x + 1 \\
\text{L3 ... } & \\
\end{align*}
\]
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 ($y \neq 0$) ?
  BEQ L2 ; $== \text{(opposite condition to } \neq \text{ in pseudo-code)}$
  MUL R0, R1, R0 ; $r = r \times x$
  SUB R2, R2, #1 ; $y = y - 1$
  B L1 ; repeat
L2 ...
```