CS1021 Tutorial #6 Partial Solution
Using Memory

1 Pseudo-code to ARM Assembly Language

```
    LDR R0, =0          ; a = 0
    LDR R2, =0          ; c = 0
    LDR R5, =4          ;

whSum
    CMP     R0, R3        ; while (a < N)
    BHS eWhSum          ; {
    MUL R4, R0, R5      ;   address = b + (a * 4)
    ADD R4, R4, R1      ;
    LDR R6, [R4]        ;
    ADD R2, R2, R6      ;   c = c + Memory.Word[address]
    ADD R0, R0, #1      ;   a = a + 1
    B   whSum           ; }
eWhSum
```

2 String Length

```
    MOV R0, #0          ; len = 0
    LDRB    R0, [R1]      ; char = Memory.Byte[adr]
whLen
    CMP R0, #0  ; while ( char != 0 )
    BEQ eWhLen          ; {
    ADD R0, R0, #1      ;   len = len + 1
    ADD R1, R1, #1      ;   add = add + 1
    LDRB    R0, [R1]      ;   char = Memory.Byte[adr]
eWhLen                  ; }
```

The following solution eliminates one of the LDRB instructions. The above solution may be easier to understand and describe in pseudo-code. Either solution is acceptable.

```
    MOV R0, #0          ; len = 0
whLen
    LDRB    R0, [R1]      ; while ( (char = Memory.Byte[adr])
    CMP R0, #0          ;         != 0)
    BEQ eWhLen          ; {
    ADD R0, R0, #1      ;   len = len + 1
    ADD R1, R1, #1      ;   add = add + 1
eWhLen                  ; }
```

There is a more efficient solution. Instead of adding one to the length of the string during each iteration of the loop, just find the address of the end of the string and subtract the address of the start of the the string from it.

3 String Duplication

As before, the singe LDRB instruction could have been replaced with two LDRB instructions, one before the while loop to load the first character and the second at the bottom of the loop to load the bext character.

```
whCpy
    LDRB    R2, [R1]     ; while ((ch = Memory.Byte[adr1])
    CMP R2, #0       ;        != NULL)
    BEQ eWhCpy       ; {
    STRB    R2, [R0]     ;   Memory.Byte[adr2] = ch
    ADD R1, R1, #1   ;   adr1++
    ADD R0, R0, #1   ;   adr2++
    B   whCpy        ; }
eWhCpy
```

4 String Reversal

```
; copy the src string pointer
    MOV R4, R1

; find the end of the src string while moving the dst pointer
; forward to create enough space to store the reversed string
whEnd
    LDRB    R2, [R1]     ; while (ch = Memory.Byte[adrSrc])
    CMP R2, #0       ;        != NULL)
    BEQ eWhEnd       ; {
    ADD R1, R1, #1   ;   adrSrc++
    ADD R0, R0, #1   ;   adrDst++
    B   whEnd        ; }
eWhEnd

; NULL-terminate the dst string
    MOV R2, #0       ; ch = NULL
    STRB    R2, [R0]     ; Memory.Byte[adrDst] = ch
    SUB R0, R0, #1   ; adrDst--

; restore the src string pointer to the start of the src string
    MOV R1, R4

; Copy the src string to the dst, moving the src string pointer
; forwards and the dst string pointer backwards
whCpy
    LDRB    R2, [R1]     ; while ((ch = Memory.Byte[adrSrc])
    CMP R2, #0       ;        != NULL)
    BEQ eWhCpy       ; {
    STRB    R2, [R0]     ;   Memory.Byte[adrDst] = ch
    ADD R1, R1, #1   ;   adrSrc++
    SUB R0, R0, #1   ;   adrDst--
    B   whCpy        ; }
eWhCpy
```