CS1021 Tutorial #6 Solution
Using Memory

1 String Length

```
1           MOV     R0, #0          ; len = 0
2           LDRB    R4, [R1]        ; char = Memory.Byte[adr]
3   whLen
4           CMP     R4, #0          ; while ( char != 0 )
5           BEQ     eWhLen          ; {
6           ADD     R0, R0, #1      ;   len = len + 1
7           ADD     R1, R1, #1      ;   add = add + 1
8           LDRB    R4, [R1]        ;   char = Memory.Byte[adr]
9           B       whLen           ; }
10  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.

```
1           MOV     R0, #0          ; len = 0
2   whLen
3           LDRB    R4, [R1]        ; while ( (char = Memory.Byte[adr])
4           CMP     R4, #0          ;         != 0)
5           BEQ     eWhLen          ; {
6           ADD     R0, R0, #1      ;   len = len + 1
7           ADD     R1, R1, #1      ;   add = add + 1
8           B       whLen           ; }
9   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.

2 String Duplication

As before, the single 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.

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

```
eWhCpy
        STRB    R2, [R0]        ; NULL terminate new string
```

3 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
```

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
```

5 Palindromes

There are very many ways to do this. Here is the program we came up with in Monday's lecture:

```
1         AREA    SandBox, CODE, READONLY
2         IMPORT  main
3         EXPORT  start
4
5 start
6
7 ;initialise strings
8
9         LDR     R1, =stringA    ; adr1 = start of string
10        MOV     R2, R1          ; adr2 = start of string
11
12 ; advance adr2 to end of string
13
14 whFindEnd                      ; while (
15        LDRB    R4, [R2]        ;   (ch = Memory.Byte[adr2])
16        CMP     R4, #0          ;     != NULL )
17        BEQ     ewhFindEnd      ; {
18        ADD     R2, R2, #1      ;   adr2++
19        B       whFindEnd       ; }
20 ewhFindEnd
21
22 ; move adr2 back from NULL to last non-NULL char
23
24        SUB     R2, R2, #1      ; adr2--
25
26 ; assume string is a palindrome ...
27
28        MOV     R0, #1          ; isPal = true
29
30 ; ... then look for non-matching characters
31
32 whPal   CMP     R0, #1          ; while (isPal
33        BNE     eWhPal          ;        &&
34        CMP     R1, R2          ;        adr1 < adr2)
35        BHS     eWhPal          ; {
36        LDRB    R5, [R1]        ;   ch1 = Memory.Byte[adr1]
37        LDRB    R6, [R2]        ;   ch2 = Memory.byte[adr2]
38        CMP     R5, R6          ;   if (ch1 != ch2)
39        BEQ     endIfDiff       ;   {
40        MOV     R0, #0          ;     isPal == false
41 endIfDiff                      ;   }
42        ADD     R1, R1, #1      ;   adr1++
43        SUB     R2, R2, #1      ;   adr2--
44        B       whPal           ; }
45 eWhPal
46
47 stop    B       stop
48
49        AREA    TestData, DATA, READWRITE
50
51 stringA
52        DCB     "kayak",0
```