CS1021 Tutorial #9 Solution
Typical Examination Questions

1 Scrabble

Approach: For each letter in the word, translate the ASCII code for the letter into an offset into the sequence of 26 score values. For example, the letter 'A' should translate into an offset of 0, the letter 'c' should translate into an offset of 2, etc..

```
score = 0;
strAdr = strStartAddress
scrAdr = scrStartAddress

while ( (char = Memory.Byte[srcAdr]) != 0)
{
    offset = char - 'A'
    charScore = Memory.Byte[scrAdr + offset]
    score = score + charScore
    srcAdr = srcAdr + 1
}
```

```
1  MOV     R0, #0          ; score = 0
2  whWord
3  LDRB    R3, [R1], #1    ; while ( (char = Memory.Byte[srcAdr])
4  CMP     R3, #0          ;         != 0 )
5  BEQ     eWhWord         ; {
6  SUB     R3, R3, #'A'    ;   offset = char - 'A'
7  LDRB    R4, [R2, R3]    ;   charScore = Memory.Byte[scrAdr + offset]
8  ADD     R0, R0, R4      ;   score = score + charScore
9  B       whWord          ; }
```

2 Increasing Sequences

Assume count is ≥ 1 and there is at least one value in the sequence.

Iterate through each value in the sequence, starting with the second value, remembering the previous value so it can be compared with the current value. Use a boolean value to indicate whether the current value falls within an increasing sequence. The start of a new sequence can be detected if the boolean is false the the current value is greater than the previous. The end of a sequence can be detected if the boolean is true and the current value is not greater than the previous.

```
result = 0

preVal = Memory.Word[adr]
adr = adr + 4
count = count - 1

increasing = FALSE

while (count != 0)
{
    curVal = Memory.Word[adr]
    adr = adr + 4
    count = count - 1

    if (!increasing && curVal > preVal)
    {
        increasing = TRUE
        result = result + 1
    }
    else if (increasing && curVal <= preVal)
    {
        increasing = FALSE
    }

    preVal = curVal
}
```

```
1     LDR         R0, #0        ; result = 0
2
3     LDR         R4, [R1]      ; load initial preVal
4     ADD         R1, R1, #4
5     SUB         R2, R2, #1
6
7     MOV         R5, #0        ; increasing = FALSE initially
8
9  whSeq
10    CMP         R2, #0        ; while (count != 0)
11    BEQ         eWhSeq        ; {
12    LDR         R3, [R1]      ;   vurVal = Memory.Word[adr]
13    ADD         R1, R1, #4    ;   adr = adr + 4
14    SUB         R2, R2, #1    ;   count = count - 1
15
16    CMP         R5, #0        ;   if (!increasing && curVal > preVal)
17    BNE         elsIfEOS      ;   {
18    CMP         R3, R4        ;
19    BLE         elsIfEOS      ;
20    MOV         R5, #1        ;     increasing = TRUE
21    ADD         R0, R0, #1    ;     result++
22    B         elfEOS          ;   }
23 elsIfEOS
24    CMP         R5, #0        ;   else if (increasing && curVal <= preVal)
25    BEQ         elfEOS        ;   {
26    CMP         R3, R4        ;
27    BGT         elfEOS        ;
28    MOV         R5, #0        ;     increasing = FALSE
29 elfEOS                       ;   }
30    MOV         R4, R3        ;   preVal = curVal
31 eWhSeq                       ; }
```