CS1021 Tutorial #10 Solution
Sample Exam Questions

(a)

```
1           MOV     R2, #1
2           MOV     R4, #1
3           MOV     R3, #1
4   fori    CMP     R3, #32
5           BHS     efori
6           AND     R1, R0, #1
7           MOV     R0, R0, LSR #1
8           AND     R5, R0, #1
9           CMP     R1, R5
10          BNE     elsneq
11          ADD     R2, R2, #1
12          B       eifeq
13  elsneq
14          CMP     R2, R4
15          BLS     eifhi
16          MOV     R4, R2
17  eifhi
18          MOV     R2, #1
19  eifeq
20          ADD     R3, R3, #1
21          B       fori
22  efori
23          CMP     R2, R4
24          BLS     eifhi2
25          MOV     R4, R2
26  eifhi2
```

(b) While the value in R0 is not less than 1010, isolate the four lest significant bits of R1 in a remporary register and compare them with 1010, adding one to the result if equal. Shift R1 right by one bit and repeat.

```
1          MOV     R1, #0
2  wh
3          CMP     R0, #2_1010
4          BLO     ewh
5          AND     R2, R0, #2_1111
6          CMP     R2, #2_1010
7          BNE     skip
8          ADD     R1, R1, #1
9  skip
10         MOV     R0, R0, LSR #1
11         B       wh
```

(c) There are a number of ways this can be done. We could count the number of As, the number of Bs, and so on. This would require 26 x n iterations but would use no additional memory. Instead, we will use 26 words in memory to store a count of the number of occurrences of each letter as we move through the string. (Beginning at the address in R3, the first word will contain our count of As, the second word will cotain the count of Bs, and so on. The idea is similar to the Scrabble problem.)

Finally, we will need to iterate through the letter counts to find the maximum value. This will be our result.

```
// We need to set every letter count to 0 first
for (i = 0; i < 26; i++) {
        Memory.Word[countaddr + (i * 4)] = 0
}

// Count the number of occurrences of each letter
while ((char=Memory.Byte[straddr]) != 0) {
        // Convert to uppercase to simplify letter check
        char = char & 0xDF   // mask is inverse of 0x20
        if (char >= A && char <= Z) {
                index = (char - 'A')
                count = Memory.Word[countaddr + (index * 4)]
                count++
                Memory.Word[countaddr + (index * 4)] = count
        }
        straddr++
}

// Find the maximum count
resLetter = 'A'
resCount = 0
for (i = 0; i < 26; i++) {
        count = Memory.Word[countaddr + (i * 4)]
        if (count > resCount) {
                resLetter = 'A' + i
                resCount = count
        }
}
```

```
1         MOV     R6, #0
2         MOV     R4, #0
3 whZero
4         CMP     R4, #26
5         BHS     eWhZero
6         MOV     R5, R4, LSL #2  ; index * 4
7         ADD     R5, R5, R3
8         STR     R6, [R5]
9         ADD     R4, R4, #1
10        B       whZero
11 eWhZero
12
13 whCount
14        LDRB    R4, [R2]
15        CMP     R4, #0
16        BEQ     eWhCount
17        BIC     R4, R4, #20
18        CMP     R4, #'A'
```

```
19          BLO     elfLetter
20          CMP     R4, #'Z'
21          BHI     elfLetter
22          SUB     R4, R4, #'A'
23          MOV     R4, R4, LSL #2  ; index * 4
24          ADD     R4, R4, R3
25          LDR     R5, [R4]
26          ADD     R5, R5, #1
27          STR     R5, [R4]
28  elfLetter
29          ADD     R2, R2, #1
30          B       whCount
31  eWhCount
32
33          MOV     R0, #'A'
34          MOV     R1, #0
35          MOV     R4, #0
36  whZero
37          CMP     R4, #26
38          BHS     eWhZero
39          MOV     R5, R4, LSL #2  ; index * 4
40          ADD     R5, R5, R3
41          LDR     R6, [R5]
42          CMP     R6, R1
43          BLS     elfHigher
44          MOV     R1, R6
45          ADD     R0, R4, #'A'
46  elfHigher
47          ADD     R4, R4, #1
48          B       whZero
49  eWhZero
```

(d) Similar to the Proper Case problem, we need to identify spaces in the original string and, when we find one, capitalise the next alphabetic character. All other characters should be lower case. We can simplify the program as we are told that the original string only contains alphabetic characters and spaces. When generating the new string, we don't store the spaces from the original string.

```
while ( (char = Memory.Byte[str1]) != NULL) {
        if (char == SPACE) {
                spaces=TRUE
        } else {
                if (spaces == TRUE) {
                        char = char & 0xDF
                        spaces = FALSE
                } else {
                        char = char | 0x20
                }
                Memory.Byte[str2] = char
                str2++
        }
        str1++
}
Memory.Byte[str2] = NULL
```

```
1  NULL     EQU     0
2  TRUE     EQU     1
3  FALSE    EQU     0
4
5          MOV     R5, #FALSE
6  whStr
7          LDRB    R4, [R1]
8          CMP     R4, #NULL
9          BEQ     eWhStr
10         CMP     R4, #' '
11         BNE     notSpaceCh
12         MOV     R5, #TRUE
13         B       elfSpaceCh
14 notSpaceCh
15         CMP     R5, #TRUE
16         BNE     notSpaces
17         BIC     R4, R4, #0x20
18         MOV     R5, #FALSE
19         B       elfSpaces
20 notSpaces
21         ORR     R4, R4, #0x20
22 elfSpaces
23         STRB    R4, [R0]
24         ADD     R0, R0, #1
25 elfSpaceCh
26         ADD     R1, R1, #1
27         B       whStr
28 eWhStr
```