

# CS1021: Assignment #2

My report on the second computing assignment is as follows:

The assignment was split into three parts: 2.1: Sets – Closure, 2.2: Sets – Symmetric Difference, 2.3: Anagrams

## 2.1: Sets – Closure

This part involved testing a set of closure i.e. every numeral element requires its opposite to close, -4 to +4.

I first started this by finding half of the size of the set. This is how many times I had to test the set to see if it was closed.

```
15 whileA
16     CMP R11, #0      ; while (remainder!=0)
17     BEQ endWh        ; {
18     SUB R11, R11, #2; {remainder = remainder-2
19     ADD R13, R13, #1; {quotient ++
20     B whileA          ; }
21 endWh
22
```

I was able to use the method above to half the set size as a condition to show that the set was closed. Every time an element was matched the element was turned to 0 to prevent rematching and a counter R5 tracked the success rate. When the two were equal the set was closed.

```
1 whileCharsChecked          ; while
2     LDR R10, =0             ; sum
3     CMP R3, R4              ; (count<#ofElemsRemaining)
4     BHS endWhileCharsChecked ; {
5
6
7     CMP R8, #0
8     BEQ endA
9     ADD R3, R3, #1          ; count = count + 1
10    ADD R1, R1, #4           ; adress=address+4
11    LDR R9, [R1]             ; charB=Memory.word[adr]
12    CMP R9, #0
13    BEQ endB
14    ADD R10, R8, R9          ; sum = charA + charB
15    CMP R10, #0             ; if(sum = 0)
16    BEQ endWhileCharsMatch   ;{
17 endB                        ; }
18
19 B whileCharsChecked         ; }
```

The while loop above takes two numbers of the elements compares one to another and if they're not equal the next number is taken. If the two numbers are a paired set the code will branch to the bottom where it gets reset. If there are no matches then the set isn't closed. If

the set contains a 0 in the middle it means it the element was already used and it will branch to increase the address and take the next number for comparison. Unfortunately due to some last minute alterations to the program I was able to create a problem I couldn't fix. Unfortunately when changing a number in the set to 0 the next number selected will be invalid.

Outputs:

```
R0 0x00000001
ASize DCD 8 ; Number of elements in A
AElems DCD +4,-6,-4,+3,-8,+6,+8,-3 ; Elements of A
```

Unfortunately when I changed the input the output was still the same.

```
R0 0x00000001
ASize DCD 8 ; Number of elements in A
AElems DCD +4,-6,-4,+3,-7,+6,+8,-3 ; Elements of A
```

## 2.2: Sets – Symmetric Difference

The symmetric difference project takes two sets compares them and put any numbers not in intersection into a new set – C. It also takes into account the amount of numbers in the size set. First I loaded all the various elements into corresponding registers. I used 2 while loops in the project to alternate between changing the element a and b.

-----	R0	0x00000017
-----	R1	0xA1000008
-----	R2	0xA1000004
-----	R3	0xA1680400
-----	R4	0xA1000028
-----	R5	0xA1000048
-----	R6	0xA1000044
-----	R7	0x00000008
-----	R8	0x00000004
-----	R9	0x00000005
-----	R10	0x00000000

Outputs

### 2.3: Anagrams

This part involved the testing of 2 words against each other and storing 1 in R0 if one word is an anagram of the other. To accomplish this I first changed each character in every word to lowercase. I took advantage of this lengthy process to count the number of letters in each word with a count. Due to my program taking even capital letters it can be put to more use.

I started off with the assumption that the two words were anagrams and I was trying to prove the assumption wrong.

```

    CMP R7, R8
    BEQ anagram
    ADD R9, R9, #1      ;count++
    ADD R2, R2, #1      ; adress=address+1
    LDRB R8, [R2]       ; charB=Memory.word[adr]
    CMP R8, #0x0        ; if (next char =0)
    BEQ notAnagram
    B while3
anagram
    ADD R8, R8, #0x20    ;char=upper ;if already used
    STRB R8, [R2]
    SUB R2, R2, R9
    LDR R9, =0

    LDRB R8, [R2]

```

It compared the first letter in both words to each other. If it didn't match the program then took the second letter from the second word. If however the two letter where the same then the letter in the second word was changed to upppercase in order to prevent the reusing of the letter. This was stored and then the adress was put back and another letter was taken in. If the next ketter in either word was null meant that it wasn't an anagram as it looped to the bottom and changed the register 0.

Tacos and coats was the sample anagram that I used as you can see below there was a 1 in register 0 to show that the two words are anagrams.

```

stringA DCB "tacos",0
stringB DCB "coats",0

```

R0 0x00000001

The default test string was bests and beets.

```

stringA DCB "bests",0
stringB DCB "beets",0

```

R0 0x00000000

I was very happy with this program as it works with various letters and number of letters. It doesn't fail when subject to vigorous testing procedure.

Register	Value
<b>Current</b>	
R0	0x00000001
R1	0xA100000A
R2	0xA100000B
R3	0x00000000
R4	0x00000000
R5	0x00000006
R6	0x00000006
R7	0x00000000
R8	0x00000086
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0xA0000361
R13 (SP)	0xA1001018
R14 (LR)	0xA0000114
R15 (PC)	0xA0000344
CPSR	0x60000010
SPSR	0x00000000
<b>User/System</b>	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
<b>Internal</b>	
PC \$	0xA0000344
Mode	User
States	20666490
Sec	0.28703458

```

66      CMP R7, R8
67      BEQ anagram
68      ADD R9, R9, #1      ;count++
69      ADD R2, R2, #1      ; address=address+1
70      LDRB R8, [R2]      ; charB=Memory.word[adr]
71      CMP R8, #0x0      ; if (next char =0)
72      BEQ notAnagram
73      B while3
74      anagram
75      ADD R8, R8, #0x20   ;char=upper ;if already used
76      STRB R8, [R2]
77      SUB R2, R2, R9
78      LDR R9, =0
79
80      LDRB R8, [R2]
81      ADD R1, R1, #1      ;adr++
82      LDRB R7, [R1]
83      CMP R7, #0x0      ;if (next char=0)
84      BEQ endcompletely ;break
85      B while3
86
87      notAnagram
88      MOV R0, #0          ; anagram=0
89
90      endcompletely
91      stop B stop
92
93
94
95      AREA TestData, DATA, READWRITE
96
97      stringA DCB "fidlle",0
98      stringB DCB "FiLLEd",0
99
100     END

```

A mix of upper and lower case letters worked as illustrated above.

0	0x00000001
1	0xA100000B
2	0xA100000C
3	0x00000000
4	0x00000000
5	0x00000007
6	0x00000007
7	0x00000000
8	0x00000086
9	0x00000000
10	0x00000000
11	0x00000000
12	0xA0000361
13 (SP)	0xA1001018
14 (LR)	0xA0000114
15 (PC)	0xA0000344
SPSR	0x60000010
SPSR	0x00000000

**System**

Interrupt

Supervisor

Undefined

Internal

\$ 0xA0000344

Mode User

States 34583191

Sec 0.48032210

```

68      ADD R9, R9, #1      ;count++
69      ADD R2, R2, #1      ; address=address+1
70      LDRB R8, [R2]      ; charB=Memory.word
71      CMP R8, #0x0      ; if (next char =0)
72      BEQ notAnagram
73      B while3
74      anagram
75      ADD R8, R8, #0x20   ;char=upper ;if alr
76      STRB R8, [R2]
77      SUB R2, R2, R9
78      LDR R9, =0
79
80      LDRB R8, [R2]
81      ADD R1, R1, #1      ;adr++
82      LDRB R7, [R1]
83      CMP R7, #0x0      ;if (next char=0)
84      BEQ endcompletely ;break
85      B while3
86
87      notAnagram
88      MOV R0, #0          ; anagram=0
89
90      endcompletely
91      stop B stop
92
93
94
95      AREA TestData, DATA, READWRITE
96
97      stringA DCB "fi dille",0
98      stringB DCB "Fi LLEd",0
99

```

Even spaces counted as shown above.