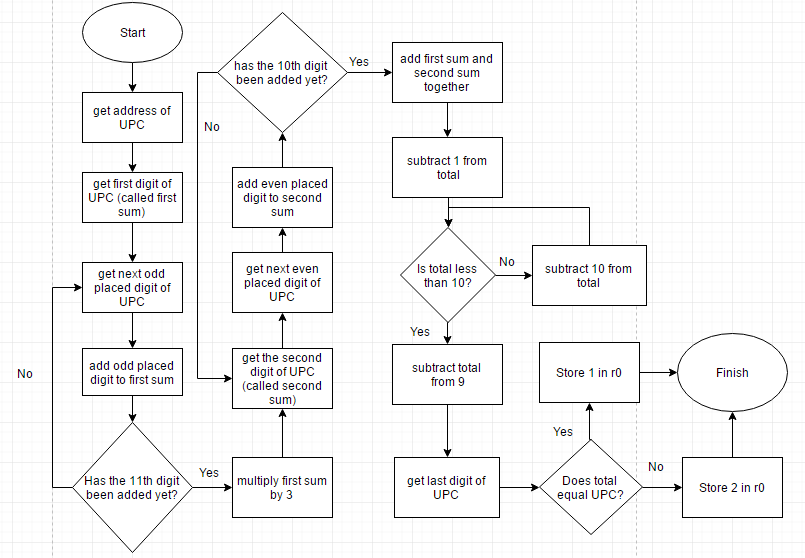
**Question 1**

Flowchart:



Code:

AREA question1, CODE, READONLY  
ENTRY  
ADR r1, UPC ; get address of UPC  
LDRB r2, [r1] ; load first byte of UPC into r2  
LDRB r4, [r1, #1] ; load second byte of UPC into r4  
SUB r2,r2, #48 ; get first digit of UPC  
SUB r4,r4, #48 ; get second digit of UPC  
MOV r12, 2 ; r12 represents the byte to be loaded (ie, initialized for the third byte)  
MOV r11, 5 ; r11 represents the number of repetitions to add a digit to the sum (there are 5 digits to be added for the first sum)

;add the 1st, 3rd, 5th, 7th ,9th and 11th digits together (sum is stored in r2)  
first LDRB r3, [r1, r12] ; get the byte that is 2 bytes ahead of r3 SUB r3, r3, #48 ; get the digit of the UPC from the byte  
 ADD r2, r2, r3 ; add to sum of r2  
 ADD r12, r12, #2 ; increment the byte to be loaded next  
 SUBS r11, r11, #1 ; decrement counter  
 BNE first ; loop until digits are added

MOV r12, 3 ; set 4th byte to be the next byte loaded  
MOV r11, 4 ; reset counter for the second sum (there are 4 values to be added)

;add the 2nd, 4th, 6th, 8th and 10th digits together (sum is stored in r4)  
second LDRB r5, [r1, r12] ; get the byte that is 2 bytes ahead of r5  
 SUB r5, r5, #48 ; get the digit of the UPC from the byte  
 ADD r4, r4, r5 ; add to sum of r4  
 ADD r12, r12, #2 ; increment the byte to be loaded next  
 SUBS r11, r11, #1 ; decrement counter  
 BNE second ; loop until digits are added

ADD r2, r2, r2, LSL #1 ; multiple first sum by 3 (r2 + r2\*2)  
ADD r2, r2, r4 ; add first and second sum  
SUB r2, r2, #1 ; subtract 1 from total sum  
MOV r10, #10 ;r10 will be used to check if the remainder from dividing by 10 has been reached yet via subtraction

;subtract 10 from r2 until remainder of r2/10 is reached  
divide SUBS r2, r2, #10 ;subtract 10 from r2  
 CMP r10, r2 ;check if 10 is greater than r2  
 BLE divide ;loop if 10 less than or equal to r2  
  
 MOV r9, #9 ;r9 will be used to subtract 9 from r2  
 SUB r2, r9, r2 ;subtract 9 from r2  
 LDRB r8, [r1, #11] ;get check byte  
 SUBS r8, r8, #48 ;get check digit

;validate the check digit  
CMP r8, r2 ;compare calculated check digit with actual check digit  
BEQ valid ;if digits are equal then go to 'valid'  
MOV r0, #2 ;if digits are not equal fall through to mark r0 as invalid  
b leave ;skip past the 'valid branch'

valid MOV r0, #1 ;mark r0 as valid if digits are equal   
leave ;continue on  
loop b loop

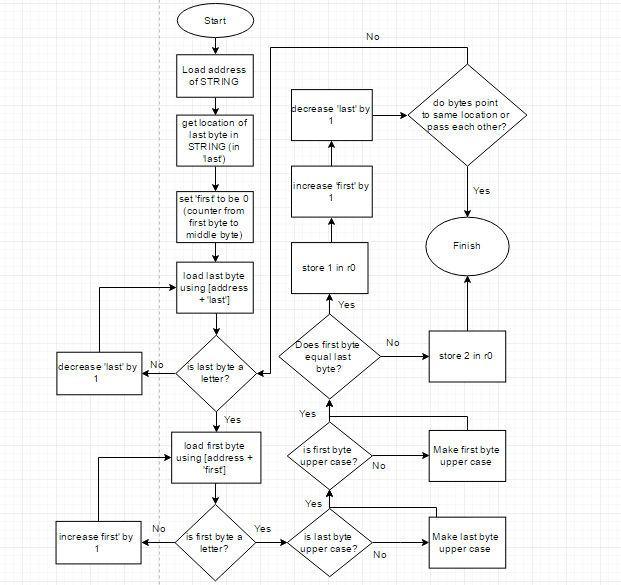
AREA question1, DATA, READWRITE  
UPC DCB "013800150738" ;UPC string  
 END

Test Data:

|  |  |
| --- | --- |
| UPC Code | Value in r0 |
| 013800150738 | 1 |
| 060383755577 | 1 |
| 065633454712 | 1 |
| 013800150739 | 2 |
| 060383755570 | 2 |
| 065633454718 | 2 |

**Question 2**

Flowchart:



Code:

AREA question2, CODE, READONLY  
ENTRY  
ADR r1, STRING ; get address of STRING  
MOV r2, 0 ; counter used to find the last byte of STRING  
MOV r5, 0 ; counter used to go from first byte to middle byte in STRING

;find location of the last byte in STRING  
last LDRB r3, [r1, r2] ;get next byte in STRING  
 CMP r3, #0 ;check if the byte is EoS  
 BEQ next ;branch to next when EoS byte is found  
 ADD r2, r2, #1 ;increment byte counter  
 b last ;loop while EoS byte not found

next SUB r2, r2, #1 ;last byte of STRING is located 1 byte less than EoS byte  
 LDRB r3, [r1] ;get first byte of STRING  
 LDRB r4, [r1, r2] ;get last byte of STRING

;check for palindrome byte by byte  
palin CMP r5, r2 ;check if bytes have crossed paths yet   
 BGE done ;if byte positions cross path exit palin

;check if r3 is a letter  
CMP r3, #65 ;check if r3 is not lower or upper case  
BLS newr3 ;get new byte if not a letter  
CMP r3, #90 ;check if r3 is lower case  
BLS goodr3 ;if r3 is lower case then byte is good, can check r4 byte next  
CMP r3, #96 ;check if r3 is in between 90 and 97  
BLE newr3 ;if r3 in in between, then r3 is not a letter, get new byte  
CMP r3, #122 ;check if r3 is upper  
BGT newr3 ;if byte is more than 122 then is not a letter, get new byte

;check if r4 is a letter  
goodr3 CMP r4, #65 ;check if r4 is not lower or upper case  
 BLS newr4 ;get new byte if not a letter  
 CMP r4, #90 ;check if r4 is lower case  
 BLS case ;if r4 is lower case then byte is good and can move on to deal with case insensitive situations  
 CMP r4, #96 ;check if r4 is in between 90 and 97  
 BLE newr4 ;if r4 in in between, then r3 is not a letter, get new byte  
 CMP r4, #122 ;check if r4 is upper  
 BGT newr4 ;if byte is more than 122 then is not a letter, get new byte  
 b case ;bytes are validated, go to case branch to deal with case insensitive situations

;gets new r3 if needed (FUNCTION)  
newr3 ADD r5, r5, #1 ;increment counter  
 LDRB r3, [r1, r5] ;get next byte  
 b palin ;go back to check for palindrome

;gets new r4 if needed (FUNCTION)  
newr4 SUB r2, r2, #1 ;decrement counter  
 LDRB r4, [r1, r2] ;get previous byte  
 b palin ;go back to check for palindrome

;deal with case insensitive situations by turning all lower into uppercase  
case CMP r3, #90 ;check if r3 is lowercase  
 BGT upperr3 ;if r3 is greater than 90 then it is lower, go to upperr3 branch   
 CMP r4, #90 ;check if r4 is lowercase  
 BGT upperr4 ;if r4 is greater than 90 then it is lower, go to upperr4 branch   
 b check ;once bytes are all upper, can go check if they are the same, go to check branch

;make r3 lower case (FUNCTION)  
upperr3 SUB r3, r3, #32 ;turn to upper case  
 b case ;go back to case branch

;make r4 lowercase (FUNCTION)  
upperr4 SUB r4, r4, #32 ;turn to upper case  
 b case ;go back to case branch

;check if letters are the same  
check CMP r3, r4 ;check if r3 == r4  
 BNE invalid ; if not equal then branch to invalid (mark r0 as invalid), else fall through to valid

;marks STRING as a valid palindrome so far  
valid MOV r0, #1 ;store 1 in r0 to mark STRING as a palindrome so far  
 SUB r2, r2, #1 ;set counter for next byte from the end  
 ADD r5, r5, #1 ;set counter for next byte from the beginning  
 LDRB r4, [r1, r2] ;get next byte from the end  
 LDRB r3, [r1, r5] ;get next byte from beginning  
 b palin ;go back to palin branch to check if new bytes are a palindrome

;marks STRING as not a palindrome (FUNCTION)  
invalid MOV r0, #2 ;store 2 in r0 to mark STRING as not a palindrome  
 b done ;can exit palin loop since it is found that STRING is not a palindrome  
done  
loop b loop

AREA question2, DATA, READWRITE  
STRING DCB "He lived as a devil, eh?" ;string  
EoS DCB 0x00 ;end of string  
 END

Test Data:

|  |  |
| --- | --- |
| String | Value in r0 |
| He lived as a devil, eh? | 1 |
| Was it a car or a cat I saw? | 1 |
| madam, I am Adam. | 2 |