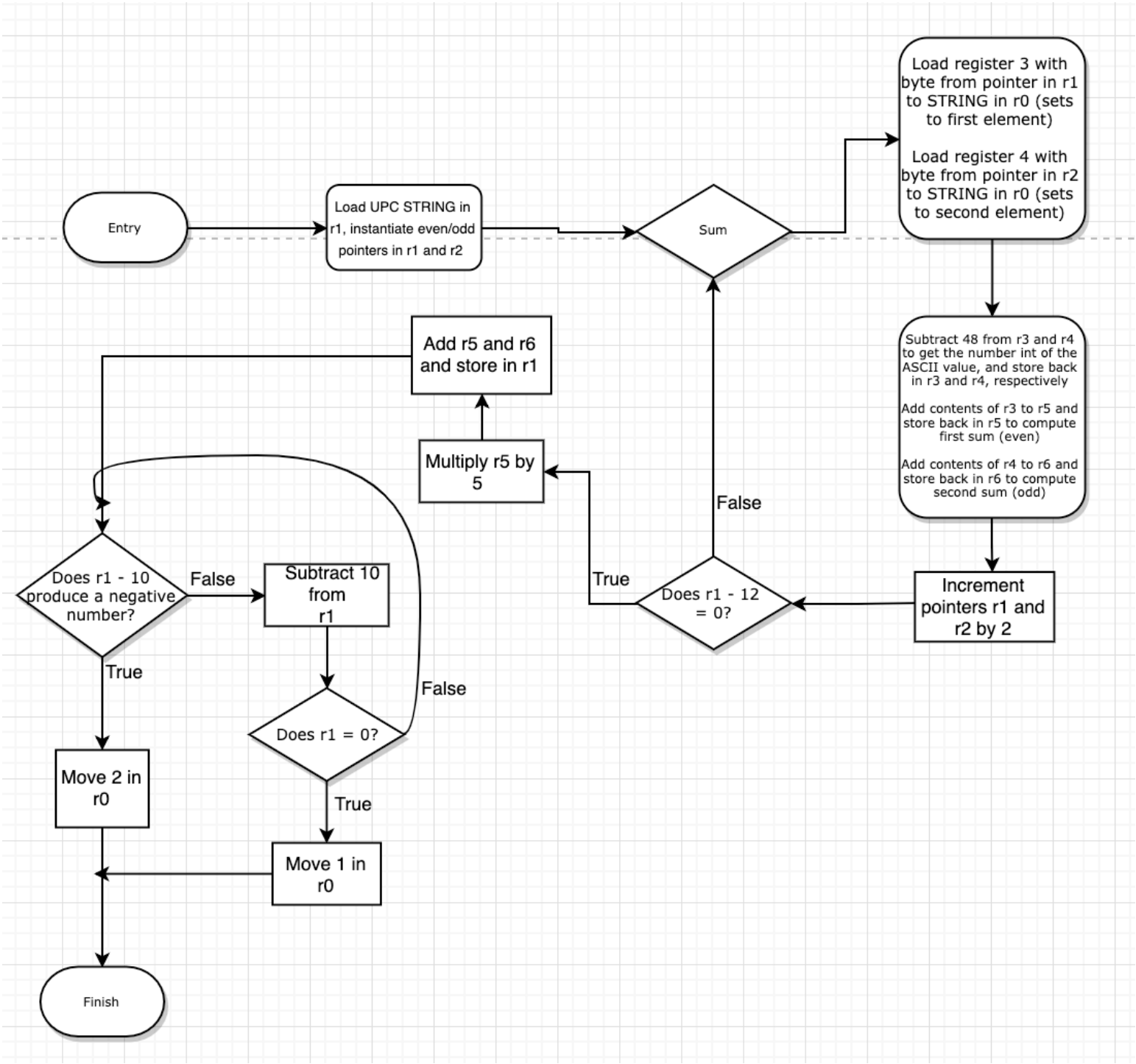
Question 1)

```
                    AREA assignment3, code, READONLY
                    ENTRY

                LDR    r0, = UPC              ; put the string from memory
location UPC into r0 to read the string
                MOV    r1, #0                 ; put number 0 into register one
to begin the pointer for even numbers
                MOV    r2, #1          ; put number 1 into register two to
begin the pointer for odd numbers

Sum             LDRB   r3, [r0, r1]    ; loads the register 3 with a byte from
the memory location at r0 pointing from register 1
                LDRB   r4, [r0, r2]    ; loads the register 4 with a byte from
the memory location at r0 pointing from register 2
                SUB    r3, r3, #48     ; takes the ASCII value from the byte
in register 3 and converts it into a number int by subtracting 48
                SUB    r4, r4, #48     ; takes the ASCII value from the byte
in register 4 and converts it into a number int by subtracting 48
                ADD    r5, r5, r3             ; complete the first sum by
adding up all of the even numbers in register 3
                ADD    r6, r6, r4             ; complete the second sum by
adding up all of the odd numbers in register 4

                ADD    r1, r1, #2             ; increment the even pointer by
2 to get the next element in the UPC string
                ADD    r2, r2, #2             ; increment the odd pointer by 2
to get the next element in the UPC String

                CMP    r1, #12                ; check to see if the counter in
register one has finished with the 10th element and is about to begin the
12th element (done)
                BNE    Sum                             ; if it is NOT equal to
0, this means it is not yet pointing to the 12th digit which means it still
has to compute digit number 10, so loop again at directive Sum

                ADD    r5, r5, LSL #1 ; multiplying the first sum by 3 using
a logical shift by one unit, and adding it to the second sum (odd numbers)
                ADD    r1, r5, r6             ; add the first sum and the
second sum together and put it in r1

Loop     CMP    r1, #10                ; compare register 1 with 10 to see if
it can be subtracted from r1 sum total
                BMI    Invalid                ; if subtracting 10 from
register one results in a negative, then we have our remainder which is less
than 10 and go to exit
                SUBS   r1, r1, #10            ; the number in register 1 is >=
10, so subtract from it and update the flags
                BEQ    Valid                  ;if the result of subtracting 10
from the register equals a 0, it means we have a remainder of 10 successful
                BNE    Loop                   ; If the result from the
previous subraction operation is a non-zero result, go back to Loop and check
again if we can keep subtracting
```

```
Valid    MOV    r0, #1                ; The computation is valid therefore
move 1 into register 0 to state so.
                B             Finish                ; Jump to finish to exit
the program

Invalid  MOV    r0, #2                ; the computation is invalid therefore
move 2 into register 0 to state so.
Finish                                                ; The jump directive
which will end the program for both valid and invalid.

UPC  DCB        "013800150738" ; UPC string

                        END
```

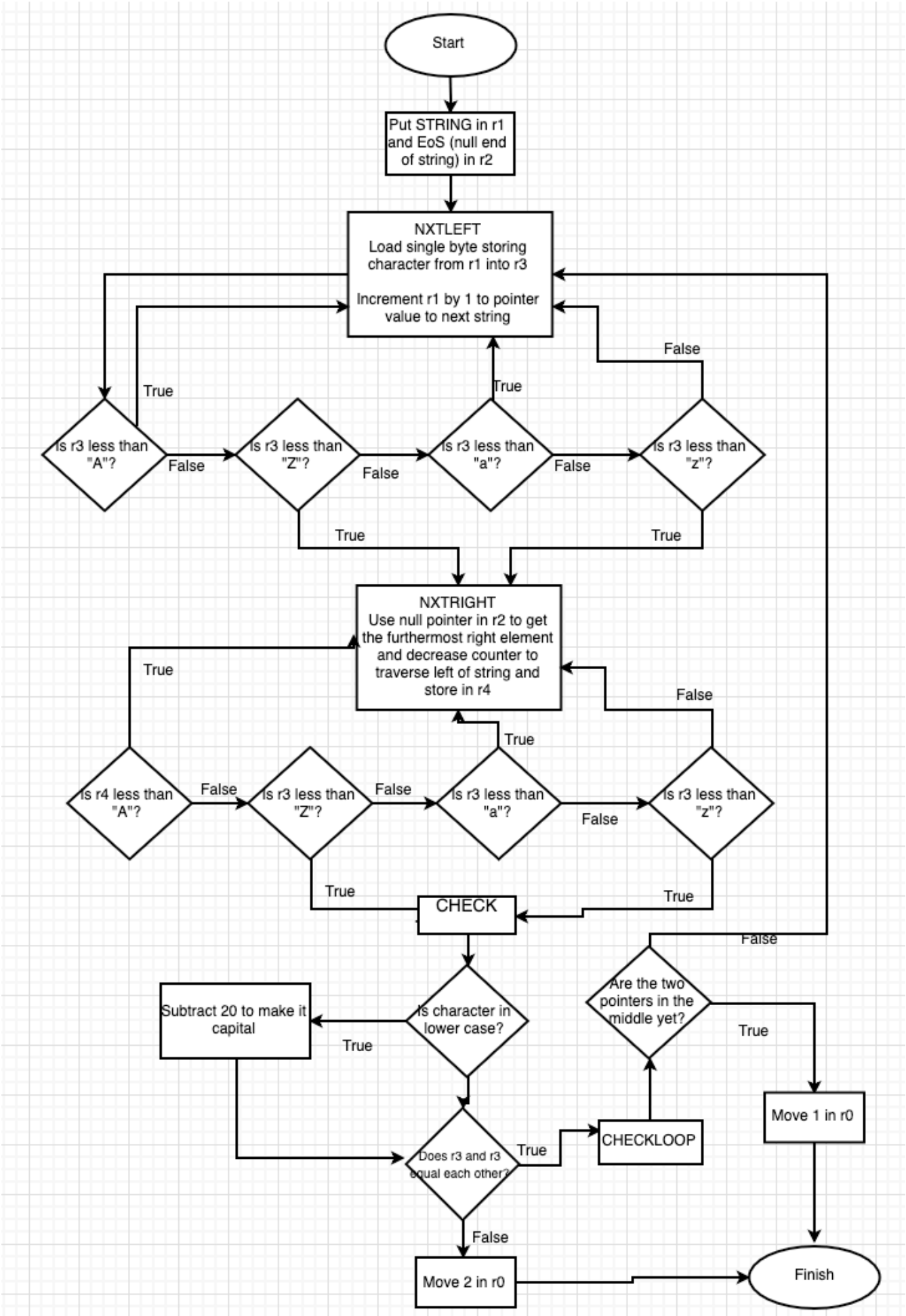

Entry
Load UPC STRING in r1, instantiate even/odd pointers in r1 and r2
Sum
Load register 3 with byte from pointer in r1 to STRING in r0 (sets to first element)
Load register 4 with byte from pointer in r2 to STRING in r0 (sets to second element)
Subtract 48 from r3 and r4 to get the number int of the ASCII value, and store back in r3 and r4, respectively
Add contents of r3 to r5 and store back in r5 to compute first sum (even)
Add contents of r4 to r6 and store back in r6 to compute second sum (odd)
Increment pointers r1 and r2 by 2
Does r1 - 12 = 0?
False
True
Multiply r5 by 5
Add r5 and r6 and store in r1
Does r1 - 10 produce a negative number?
False
Subtract 10 from r1
Does r1 = 0?
False
True
True
Move 2 in r0
Move 1 in r0
Finish

Question 2)

```
                                AREA prog2, code, READONLY
                                ENTRY

                                ADR     r1, STRING
        ;generate an address to STRING in register 1 to allow the pointers to
use it when receiving the ASCII characters (bytes)
                                ADR     r2, EoS
        ;generate an address to EoS (end of the string) in register 2 for a
label

NXTLEFT                 LDRB    r3, [r1]
        ;load a single byte storing the character from STRING in r1 to r3 to
allow for the pointer
                                ADD             r1, #1
                ;load the pointer with 1 to check the next byte in the string.
This will keep increasing in each iteration.

                                CMP             r3, #0x41
                ;compare the ASCII letter in register 1 to "A" to see if it is
below or above it
                                BLT             NXTLEFT
                ;the letter in r3 is smaller than "A" then branch to NXTLEFT
to get a new character (it is not an acceptable letter)

                                CMP             r3, #0x5A
                ;compare the ASCII letter in register 1 to "Z" to see if it is
"A" < letter < "Z" which makes it acceptable
                                BLT             NXTRIGHT
                ;the letter is between A-Z which is acceptable so get the
second letter from NXTRIGHT to compare with the other end as well

                                CMP             r3, #0x61
                ;compare the ASCII letter in register 1 to "a" to see if it is
"Z" < letter < "a"
                                BLT             NXTLEFT
                ;the ASCII letter is between Z-a which is an unnacceptable
letter so get a new one

                                CMP             r3, #0x7A
                ;compare the ASCII letter in register 1 to "z" to see if it is
"a" < letter < "z"
                                BLT     NXTRIGHT
                                B               NXTLEFT
                ;the leter is not in our defined parameters which means it is
> "z", so it is unacceptable - fetch a new one

NXTRIGHT                LDRB    r4, [r2, #-1]!                          ;load
register 7 with the pointer starting from the end of the string and moving
left

                                CMP             r4, #0x41
                ;compare the ASCII letter in register 4 to "A" (r4 - "A")
                                BLT             NXTRIGHT
                ;if the letter is smaller than "A" now get a new character
```

```
                                CMP     r4, #0x5A
        ;compare the ASCII letter in register 4 to "Z" to see if it is "A" <
letter < "Z"
                                BLT             CHECK
                ;the letter is valid so branch to check to see if r3 = r4

                                CMP             r4, #0x61
                ;compare the ASCII letter in register 4 to "a" to see if it is
"Z" < letter < "a"
                                BLT             NXTRIGHT
                ;the letter is in unnacceptable format so fetch a new one
through NXTRIGHT

                                CMP             r4, #0x7A
                ;compare the ASCII letter in register 4 to "z" to see if it is
"a" < letter <"z"
                                BLT             CHECK
                ;If the letter is less than z, it means it is any lower case
letter which is acceptable, now branch to CHECK to see if r3 = r4
                                B               NXTRIGHT
                ;the letter is not in our defined parameters which means it is
> "z", fetch a new one

CHECK                   CMP             r3, #0x61
        ;compare letter "a" with ASCII letter in register 3 which will tell us
in the next line if it is greater than a (a lower letter)
                                SUBGT   r3, r3, #0x20
        ;the letter is lower case, subtract 20 to make it capital so that we
can make evrything case insensitive

                                CMP             r4, #0x61
                ;compare letter "a" with ASCII letter in register 4 which will
tell us in the next line if it is greater than a (a lower letter)
                                SUBGT   r4, r4, #0x20
        ;the letter is lower case so subtract it by 20 to get it's capital
equivalent to make everything case insensitive

                                CMP             r3, r4
                ;compare the two letters in registers 3 and 4 to see if they
equal each other or not which will tell us if they are a palindrome
                                BEQ             CHECKLOOP
                ;If they equal each other (r3 - r4 = 0) then there is a
palindrome go to CHECKLOOP to see if we have checked all the letters
                                B               NOTPD
                ;They do not equal each other, branch to NOTPD to signify it
is not a palindrome and end the program

CHECKLOOP               CMP             r1, r2
        ;compare registers 1 and 2 (r1-r2) which store pointers to the
left/right side of the string to see if they have reached the middle yet
                                BGE             ISPD
                ;the pointers have reached the same value, now go to DONE to
move 0 in r1 and finish
                                B               NXTLEFT
                ;the pointers still have not reached the middle as they are
```

```
not equal to each other - branch to NXTLEFT to get the next iteration of
letters

ISPD                    MOV             r0, #1
        ;since the string is a palindrome and we have already checked all of
the bytes, move 1 into register 0
                                B               FINISH
                ;the program is complete, branch to FINISH to end
NOTPD                   MOV             r0, #2
        ;the string is NOT a palindrome, move 2 in register 0 and end the
program via FINISH
FINISH

STRING                  DCB     "He lived as a devil, eh?"  ;string
EoS                     DCB     0x00
        ;end of string

                                END
```


Start
Put STRING in r1 and EoS (null end of string) in r2
NXTLEFT
Load single byte storing character from r1 into r3
Increment r1 by 1 to pointer value to next string
Is r3 less than "A"?
Is r3 less than "Z"?
Is r3 less than "a"?
Is r3 less than "z"?
True
False
NXTRIGHT
Use null pointer in r2 to get the furthermost right element and decrease counter to traverse left of string and store in r4
Is r4 less than "A"?
Is r3 less than "Z"?
Is r3 less than "a"?
Is r3 less than "z"?
CHECK
Are the two pointers in the middle yet?
Is character in lower case?
Subtract 20 to make it capital
Does r3 and r3 equal each other?
CHECKLOOP
Move 1 in r0
Move 2 in r0
Finish