Assignment 4
Subroutines

Deadline: 18 March 2022 at 23:59 on Submitty

Working individually, complete the assignment below. Submit your solution to Submitty (`https://submit.scss.tcd.ie`). By submitting your solution, you are confirming that you have familiarised yourself with College's policy on plagiarism (`https://libguides.tcd.ie/plagiarism`).

Your mark will be the auto-graded mark assigned by Submitty (10 marks).

You are allowed to submit five attempts for the assignment without penalty. Subsequent attempts will attract a 1 mark penalty each, up to a maximum penalty of 5 marks.

Submitty will allow you eight "late days" over the full semester. This means, for example, you can submit one assignment late by eight days or eight assignments late by one day (or part thereof) each, without penalty. Once your "late days" are used up, you will receive zero marks for any late submissions.

Instructions

Write four ARM Assembly Language subroutines that implement the interfaces described below. Develop your solution using the `asmt-4-subroutines` template program in the CSU1102x repository. Submit your solution to Submitty (`https://submit.scss.tcd.ie`).

Note that subroutines.s does not contain a "Main" program and should only contain your subroutines, which will be called by test.s

get9x9

```
@ get9x9 subroutine
@ Retrieve the element at row r, column c of a 9x9 2D array
@   of word-size values stored using row-major ordering.
@
@ Parameters:
@   R0: address - array start address
@   R1: r - row number
@   R2: c - column number
@
@ Return:
@   R0: element at row r, column c
```

set9x9

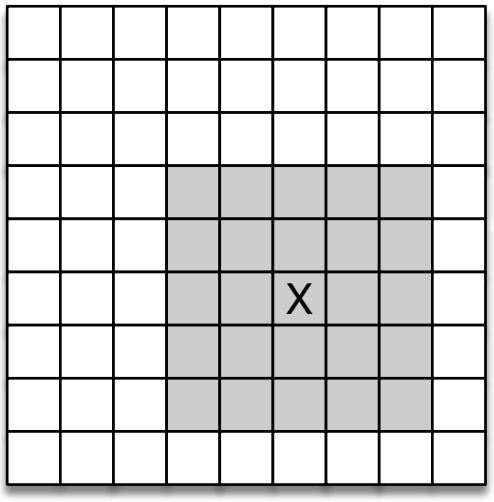
```
@ set9x9 subroutine
@ Set the value of the element at row r, column c of a 9x9
@   2D array of word-size values stored using row-major
@   ordering.
@
@ Parameters:
@   R0: address - array start address
@   R1: r - row number
@   R2: c - column number
@   R3: value - new word-size value for array[r][c]
@
@ Return:
@   none
```

average9x9

```
@ average9x9 subroutine
@ Calculate the average value of the elements up to a distance of
@   n rows and n columns from the element at row r, column c in
@   a 9x9 2D array of word-size values. The average should include
@   the element at row r, column c.
@
@ Parameters:
@   R0: address - array start address
@   R1: r - row number
@   R2: c - column number
@   R3: n - element radius
@
@ Return:
@   R0: average value of elements
```

In other words, this subroutine should calculate the average value of the elements of a (2n+1)x(2n+1) subarray centered on the element at row r, column c.

In the figure below, the shaded elements should be included in the calculation of the average of the element marked X at a radius of n=2 rows and columns.



When calculating the average for elements within r rows or n columns of the edges of the array, the average only includes the elements inside the array.

Use the UDIV instruction to find the average and ignore any fractions or rounding.

Your implementation of average9x9 MUST make use of the get9x9 subroutine that you have implemented.

blur9x9

```
@ blur9x9 subroutine
@ Create a new 9x9 2D array in memory where each element of the new
@ array is the average value the elements, up to a distance of n
@ rows and n columns, surrounding the corresponding element in an
@ original array, also stored in memory.
@
@ Parameters:
@   R0: addressA - start address of original array
@   R1: addressB - start address of new array
@   R2: n - radius
@
@ Return:
@   none
```

Your implementation of blur9x9 MUST make use of the average9x9 and set9x9 subroutines that you have implemented.