CPSC 359 – Fall 2016
Assignment 2
ARM Warmup
25 points; 4% weight
Due: October 21st @ 11:59PM (midnight)

Objective: To get started with the Raspberry Pi environment and ARM assembly programming

Outcome: To implement using the Raspberry Pi's UART interface a text based ARM assembly program that collects numbers from the user, sorts them, and calculates the median

Program Steps:

1. Start with displaying the creator name(s).

2. Your program must first collect the length of the number list to be provided (1-9).
   - Check if the input is a number between 1 and 9. If not, then display an appropriate error message as shown in the example below.

3. Loop over the list of the numbers to get each number [0-100].
   - Check if the input number is a number and between 0 and 100. Otherwise, display an appropriate error message

4. Display the sorted list and the median of the numbers.

5. Loop back to step 2.

6. The program exists when the user enters 'q' or 'Q'

Example session:

```
Created by: John Smith and Sarah Smith
Please enter the size of the number list:
>4
Please enter the first number:
>85
Please enter the second number:
>6a
Wrong number format!
Please enter the second number:
>65
Please enter the third number:
>150
Invalid number! The number should be between 0 and 100
Please enter the third number:
>90
Please enter the fourth number:
>10
The sorted list is: 10 65 85 90
The median is: 75
####################
Please enter the size of the number list:
>0
Invalid number! The size of number list should be between 1 and 9
Please enter the number of students:
>
```

UART: UART stands for Universal Asynchronous Receiver Transmitter. It is a serial communication protocol between a CPU core (e.g., on the Pi) and a low throughput I/O device, such as a tty terminal. The Pi also supports a slimmed down version of UART called mini-UART. The Pi communicates using UART through its GPIO (General Purpose I/O) headers. Initializing and using UART requires quite a bit of programming. In this assignment, you are only required to use UART as a black-box. We are providing with this assignment an object file (.o) that does all the UART work.

Notes:

1. Using the UART object file (from D2L), you can call the following subroutines to communicate over UART:
    a. **WriteStringUART**: This function is used to write a string to the UART line. That is, your program outputs a string to the tty terminal.
        - arguments:
            1. R0: String pointer.
            2. R1: Length of the string to be written.
    b. **ReadLineUART**: This function is used to read from UART until a <newline> is encountered. Input characters read from the UART must be saved in a buffer created by you.
        - arguments:
            1. R0: Address of the buffer.
            2. R1: Buffer length.
        - Return value:
            1. R0: number of ASCII characters read.
2. You will read and write ASCII characters from/to the UART interface. For reading or writing **numbers**, you need to convert ASCII characters to numbers after reading and convert numbers to ASCII characters before writing. For example, character '0' has an ASCII value of 48.
3. You can connect to the Pi over UART Line from Linux by opening a new terminal and using the following command: ***screen /dev/ttyUSB0 115200***
4. If the ***screen*** command does not work, you will need to unplug / re-plug the USB to serial link (the large USB connector plugged into the monitor stand).

Grading:

1.	Display creator names:	1
2.	Input Description	2
3.	Perform sorting	5
4.	Calculate median	3
5.	Error messages	2
6.	Converting ASCII to numbers	4
7.	Converting numbers to ASCII	4
8.	Loop back	1
9.	Efficient code	2
10.	Well documented code	1
	Total	25 points

Programs that do not compile cannot receive more than **5 points**. Programs that compile, but do not implement any of the functionality described above can receive a maximum of **7 points**.

Teams: You are advised to work with another student in class in order to complete the assignment, but you are not required to do so. You and your partner must be in tutorials taught by the **same TA**. Peer evaluation in teams may be conducted.

Submission: Submit a .tar.gz file named **c359-<student1_id-student2_id>-a2.tar.gz** of your entire project directory (including source code, make file, build objects, kernel.img, etc) to your TA via the appropriate dropbox on Desire2Learn. Only one submission per team is required.

Late Submission Policy: Late submissions will be penalized as follows:
-12.5% for each late day or portion of a day for the first two days
-25% for each additional day or portion of a day after the first two days
Hence, no submissions will be accepted after 5 days (including weekend days) of the announced deadline.

Academic Misconduct: Any similarities between assignments will be further investigated for academic misconduct. While you are encouraged to discuss the assignment with your colleagues, your final submission must be your own original work. Any re-used code of excess of 20 lines (20 assembly language instructions) must be cited and have its source acknowledged. Failure to credit the source will also result in a misconduct investigation.

D2L Marks: Any marks posted on D2L or made available using any other mean are tentative and are subject to change (after posting). They can go UP or DOWN due to necessary corrections.