CPSC 359 – Winter 2017
Assignment 2
ARM Warmup
Due: February 10th, 2017 @ 11:59
Weight: 4% of the total mark

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

Outcome: To implement using the Raspberry Pi's UART interface an ARM assembly program that allows drawing object using asterisks, calculate total number of asterisks, and mean of asterisks grouped by object.

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 supports UART. It also supports a slimmed-down version of UART called miniUART. 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.

Program Steps:

1. Start with displaying the name(s) of creator(s).
2. Your program will first display a numbered list of objects that can be drawn. There will be three objects: square, rectangle, and triangle.
3. User should select an object to draw, by selecting the object's number.
    a. Check if the input is a number between 1 and 3. If not, display an appropriate error message.
    b. If the user enters -1, then go to step 7.
    c. If the user enters q, then go to step 9.
4. Ask for object's width
    a. Check if the input is a number between 3 and 9. If not, then display an appropriate error message.
5. Draw the selected object.
6. Go to step 3.
7. If the user enters -1, then your program should display a summary as follows:
    a. Total number of asterisks used to draw all objects in the current session.
    b. Mean of asterisks used to draw all square(s).
    c. Mean of asterisks used to draw all triangle(s).
    d. Mean of asterisks used to draw all rectangle(s).
8. Got to step 2.
9. Program exits when the user enters q.

Example session:

```
Created By: John Smith and Sarah Smith
Please enter the number of object you want to draw. Press -1 for Summary or q to exit
1- Square; 2- Rectangle; 3- Triangle
1e
Wrong number format! q is only allowed as character to exit
Please enter the number of object you want to draw. Press -1 for Summary or q to exit
1- Square; 2- Rectangle; 3- Triangle
1
Please enter width of object. Please make sure it is between 3 and 9.
4
* * * *
* * * *
* * * *
* * * *
Please enter the number of object you want to draw. Press -1 for Summary or q to exit
1- Square; 2- Rectangle; 3- Triangle
7
Invalid number! the number should be between 1 and 3 or -1 for summary
Please enter the number of object you want to draw. Press -1 for Summary or q to exit
1- Square; 2- Rectangle; 3- Triangle
2
Please enter width of object. Please make sure it is between 3 and 9.
11
Invalid number! the width should be between 3 and 9
Please enter width of object. Please make sure it is between 3 and 9.
8
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
Please enter the number of object you want to draw. Press -1 for Summary or q to exit
1- Square; 2- Rectangle; 3- Triangle
-1
Total Number of Stars is: 64
Mean of Stars used to draw Square(s): 0.2
Mean of Stars used to draw Rectangle(s): 0.7
Mean of Stars used to draw Triangle(s): 0
Please enter the number of object you want to draw. Press -1 for Summary or q to exit
1- Square; 2- Rectangle; 3- Triangle
q
Terminate Program.
```

Notes:

- **Width** is used to determine the number of asterisks for width
- For a **Rectangle object** the **Height** will be **Width – 2**.
- The algorithm for drawing **Triangle(s)** can be found here: https://www.codingunit.com/c-tutorial-a-star-pyramid-and-string-triangle-using-for-loops.
- For **Triangle(s)** and **Square(s)**, you only need the **Width** value to draw them. For a triangle object, **Width** is the width of the base.

- When prompting for **Width,** the user should not be allowed to enter -1 nor q.

UART 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.
        i. arguments:
            1. R0: String pointer
            2. R1: Length of the string to be written
        ii. There is no return value for this function
    b. **ReadLineUART**: This function is used to read from UART the specified number of characters (Buffer length). Input characters read through UART must be saved in a buffer created by you.
        i. arguments:
            1. R0: Address of the buffer.
            2. R1: Buffer length (this is the maximum number of characters that can be read).
        ii. Return value in R0: the number of ASCII actual 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 you need to 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:

• Displaying creator names	1
• Input Description	2
• Drawing Objects	5
• Calculating mean	3
• Calculating total	2
• Error messages	2
• Converting ASCII to numbers	4
• Converting numbers to ASCII	4
• Looping back	1
• Efficient code	2
• Termination	1
• Displaying summary	2
• Well documented code	1

Total	**30 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 your source code to the drop box on D2L. 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 assignment submissions will be further investigated for potential academic misconduct.

Code sharing with a different group is prohibited. Code sharing includes looking at others' code on paper and on the computer screen. Discussions with other groups can only be carried out at the concept level. While you are encouraged to discuss the assignment with your colleagues, your final submission must be your team's original work. Any re-used code in excess of 10 lines must be acknowledged and cited. Violation of this policy may be considered academic misconduct. If unsure, always check with your instructor or TA.

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.