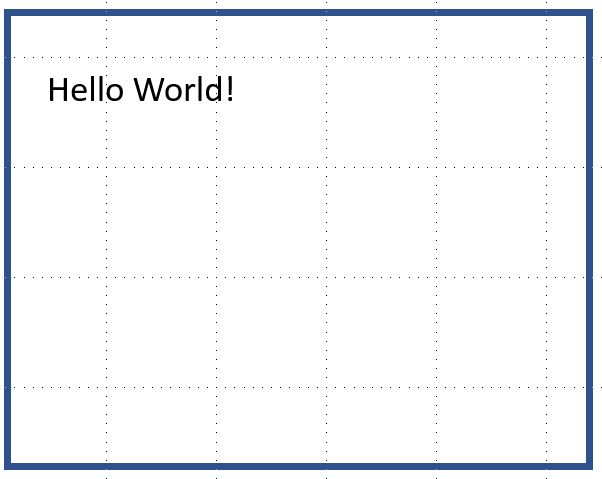
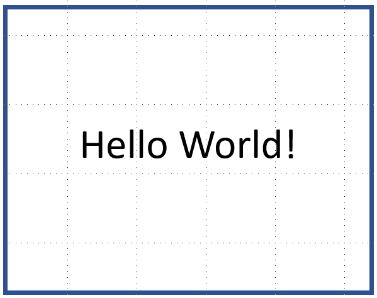
1. Q1. (10 marks) Develop the following programs to display text on the EV3 LCD.

a. Write a RobotC program to display “Hello World!” on the EV3 brick LCD.

Use displayTextLine function to display the text. Use sleep function to make the message visible for 10 seconds (Figure 1).



*Figure 1: Hello world!*

*Figure 2: Hello world (big text)*

1

* 1. Write the above program as a function a called helloWorld() and call the function from the main task.
  2. Use for loop to blink (on and off) the “Hello World!” message 5 times with 1 second intervals (i.e., message displayed for 1 second and message cleared for 1 second). Use sleep() function for timing. This time use displayCenteredBigTextLine() function to display the message. Use the helloWorld() function you developed in Q1-b (Figure 2).

**Hint:** You may define a function that prints an empty string on the same line.

* 1. Modify the functions in Q1-c, so that we can pass a parameter to change the rate of blinking. Progressively reduce the interval and determine the rate at which you can blink without any issues.

Q2. (10 marks) Multiline information display.

* 1. Write a RobotC function to compute and display the square of numbers from 1 to

8. Use displayTextLine() function to print values and sleep() function to maintain the display content for 10 seconds. Each value should occupy its own line, and all the values from 1 to 10 should be visible on the LCD (Figure 3). **Hint:** You may use breakpoints to debug and see how the variables change.

* 1. Modify the function developed in Q2-a such that display only one line at a time. After 1 second, the next line will be displayed. Each number should occupy its own line.  
     **Hint:** You may use erase display to clear the text (Figure 4).

Q3. (10 marks) The 𝑛!" Fibonacci number is defined by the recurrence relation, 𝐹 = 𝐹 + # #$%

𝐹 where 𝐹 = 0 and 𝐹 = 1. Write a RobotC program to compute the first 20 Fibonacci #$& ' %

numbers using standard (using a for or while loop) and recursive C functions. Use an integer array to store the numbers. Use displayCenteredBigTextLine() to display the content in the middle of the LCD. Use sleep() function to keep one value for 1 sec before displaying the next value. Upon reaching the last number, the program should restart again from 𝐹 and run forever. **Discuss the advantages and disadvantages of recursive**

**programming in systems with low memory and computational power**. 2

|  |  |
| --- | --- |
| *Figure 3: Square of numbers* | *Figure 4: Square of numbers version 2* |

'

Q4. (15 marks) The robot can execute a “point turn” when the wheels are commanded to turn in the opposite direction at the same speed. Write a function pointTurn() that takes as an input the amount of time to sleep. Your function should execute a counter-clockwise point turn at 20 power level using the built-in function setMotorSpeed().

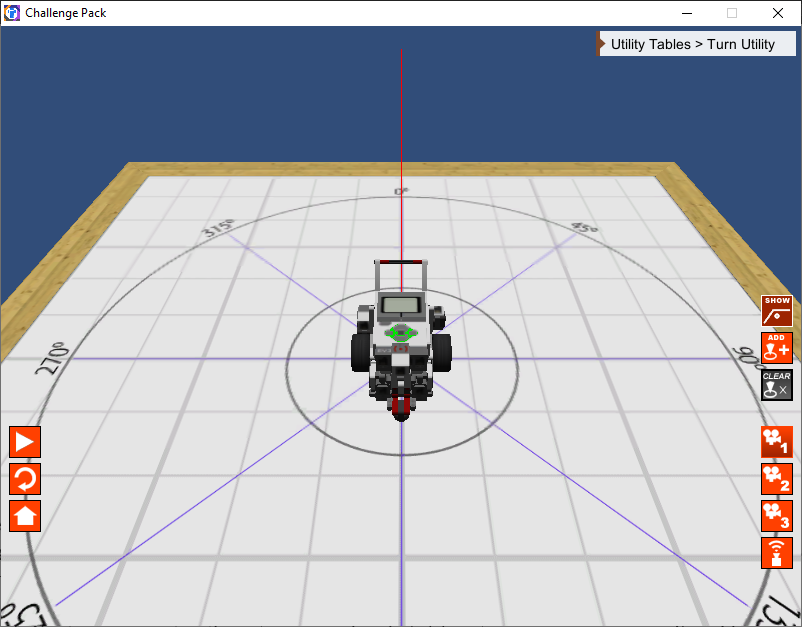
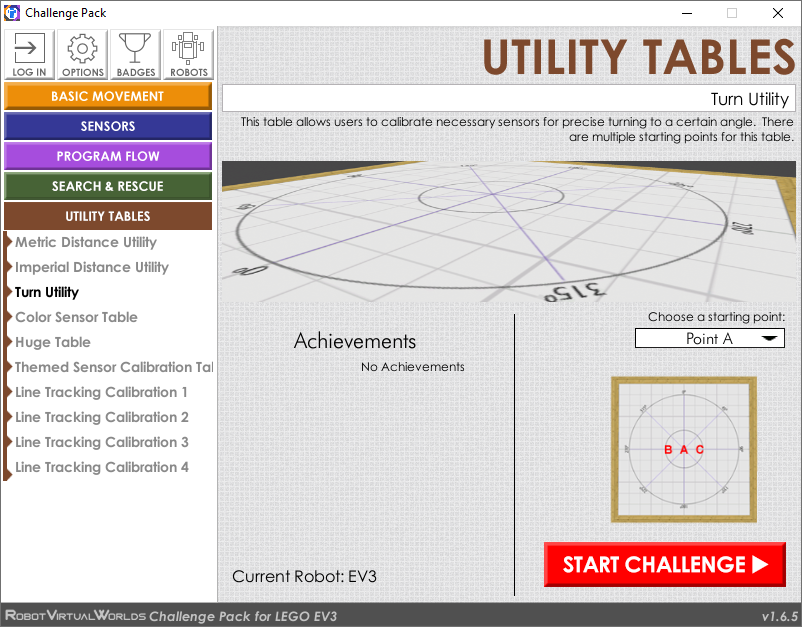
* 1. Find the amount of time it needs to turn 360 degrees.  
     Use the Turn Utility table under Utility tables in RVW. See Figure 4. Turn on the orientation tracker (Figure 6) and top-down view for better visualization.
  2. Compute the time it needs to sleep to turn 5 degrees.
  3. Based on your solution in Q4-b, verify if the robot executes precise turns 30, 45,

90, and 135 degree turns.

*Figure 5: Turn utility in RVW*

*Figure 6: Turn on orientation tracker*

3



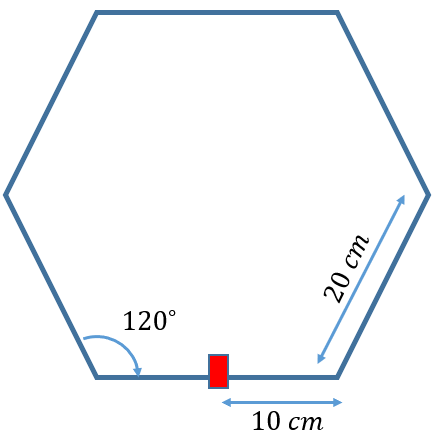
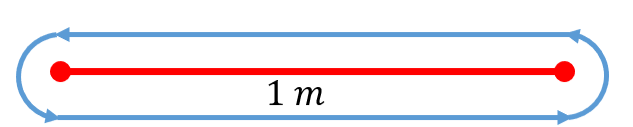
* 1. Q5.  (15 marks) You can move your robot forward (without turning) when you command the wheels to turn in the same direction at the same power level. Write a function moveStraight() that takes as an input the amount of time to sleep. Your function should execute forward motion at 50 power level using the built-in function setMotorSpeed().
     1. Iteratively test and find the amount of time it needs to move 1 m.  
        Use the **Metric Distance Utility** table under **Utility Tables** in RVW. Turn on the top-down view for better visualization.
     2. Compute the time it needs to sleep to move 5 cm.
     3. Based on your solution in Q5-b, verify if the robot executes precise 15, 50, 95,

and 150 cm movements.

* 1. Q6.  (20 marks) Develop a RobotC program to make the robot move back and forth 5 times along a straight line of length 100 cm (see Figure 7). At the end of the line, the robot should make a 180-degree counterclockwise point turn. At the beginning of the program, display “Start of the program”. During the robot motion, display on the Remote LCD the number of times it is on at any given time. In the end, display “End of the Program”.

*Figure 7: Straight line track*

* 1. Q7.  (20 marks) Develop a RobotC program to make the robot move in a hexagonal shape (see Figure 8 for dimensions) 3 times. Use the Huge Table under Utility Tables of RVW. At the end of the traversal, re-orient the robot to the starting orientation. At the beginning of the program, display “Start of the program”. During the robot motion, display on the Remote LCD the number round it is on at any given time. In the end, display “End of the Program”.



*Figure 8: Hexagonal track*

4