**W2 -** PRACTICE

*Functions & Modular Design (Part 1)*

## *At the end of this practice, you should be able to…*

* Decompose a **larger problem into smaller**, reusable functions.
* **Define and implement functions** in C with appropriate **parameters** and **return** types.
* Use **function calls** effectively Use **debugging** **techniques** to check for errors.

## *How do we structure exercises?*

We organize each practice into 3 parts:

|  |  |
| --- | --- |
| ANALYSE | **Understand** existing codes, find the **bugs** or **complete** missing gaps |
| MANIPULATE | Ensure you can **apply the theory** with some basic challenges |
| CREATE | **Express your creativity** with more complex challenges |

## A black background with a black square AI-generated content may be incorrect.*How to submit?*

* Submit this document in **UMS (LMS)**
* File format: **Fullname\_W2\_Practice.pdf**
  + *Example: Chan\_Dara\_W2\_Practice.pdf*
* Deadline: 02nd May 2025, 11:59 pm
* The score will be recorded as a **class participant** in Week 2.

## *Are you lost?*

You can read the following documentation to be ready for this practice

<https://pseudocode.deepjain.com/guides/functions/>

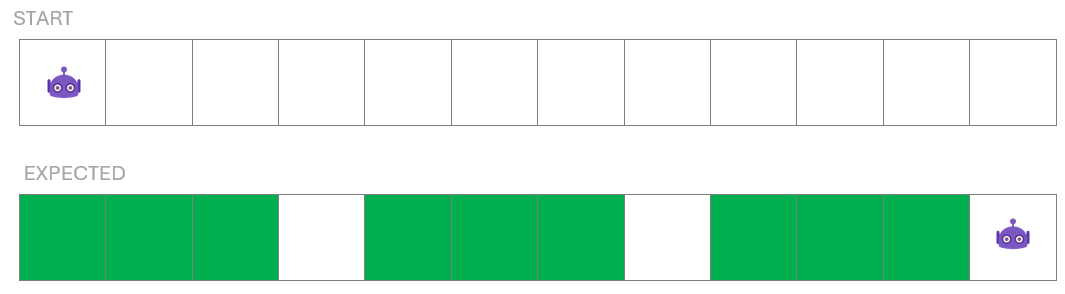
<https://www.w3schools.com/c/c_functions.php>

ANALYSE

**EX 1 (Draw lines)**

The robot needs to paint green lines.

To avoid duplicated code, we use a function. *Complete the function to fulfill the requirements.*



Available operations:

* goRight() // Move the robot to the cell on the right
* paintCell(color) // paint the robot cell with given color
* RERPEAT <x> TIMES, WHILE <c>
* IF <c>, ELSE IF <c> , ELSE

MAIN

drawLine();

goRight();

drawLine();

goRight();

drawLine();

goRight();

FUNCTION drawLine() {

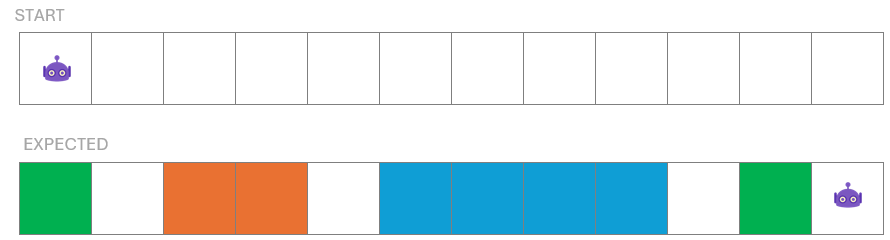
// Your code

}

**EX 2 (Draw lines of different sizes and colors)**

Now the robot needs to paint lines which can have different sizes and color.

To avoid duplicated code, we use a function. *Complete the function and main to fulfill the requirements.*



Available operations:

* goRight() // Move the robot to the cell on the right
* paintCell(color) // paint the robot cell with given color
* RERPEAT <x> TIMES, WHILE <c>
* IF <c>, ELSE IF <c> , ELSE

MAIN

drawLine( /\* complete with parameters if needed \*/ );

drawLine( /\* complete with parameters if needed \*/ );

drawLine( /\* complete with parameters if needed \*/ );

drawLine( /\* complete with parameters if needed \*/ );

FUNCTION drawLine( /\* complete with parameters if needed \*/ ) {

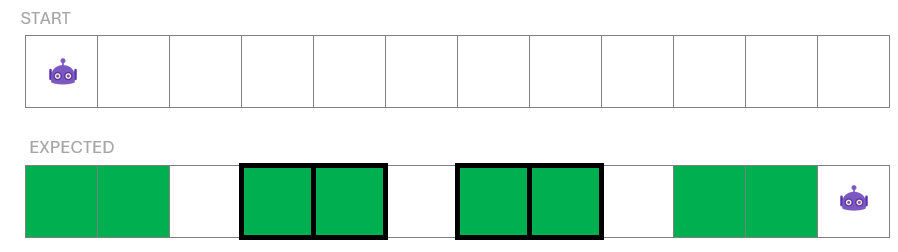
// Your code

}

**EX 3 (Draw lines with or without borders)**

Now the robot needs to paint greens lines that can have borders or not.

To avoid duplicated code, we use a function. *Complete the function and main to fulfill the requirements.*



Available operations:

* goRight() // Move the robot to the cell on the right
* paintCell(color) // paint the robot cell with given color
* paintBorder(color) // paint the robot cell with given color
* RERPEAT <x> TIMES, WHILE <c>
* IF <c>, ELSE IF <c> , ELSE

MAIN

drawLine( /\* complete with parameters if needed \*/ );

drawLine( /\* complete with parameters if needed \*/ );

drawLine( /\* complete with parameters if needed \*/ );

drawLine( /\* complete with parameters if needed \*/ );

FUNCTION drawLine( /\* complete with parameters if needed \*/ ) {

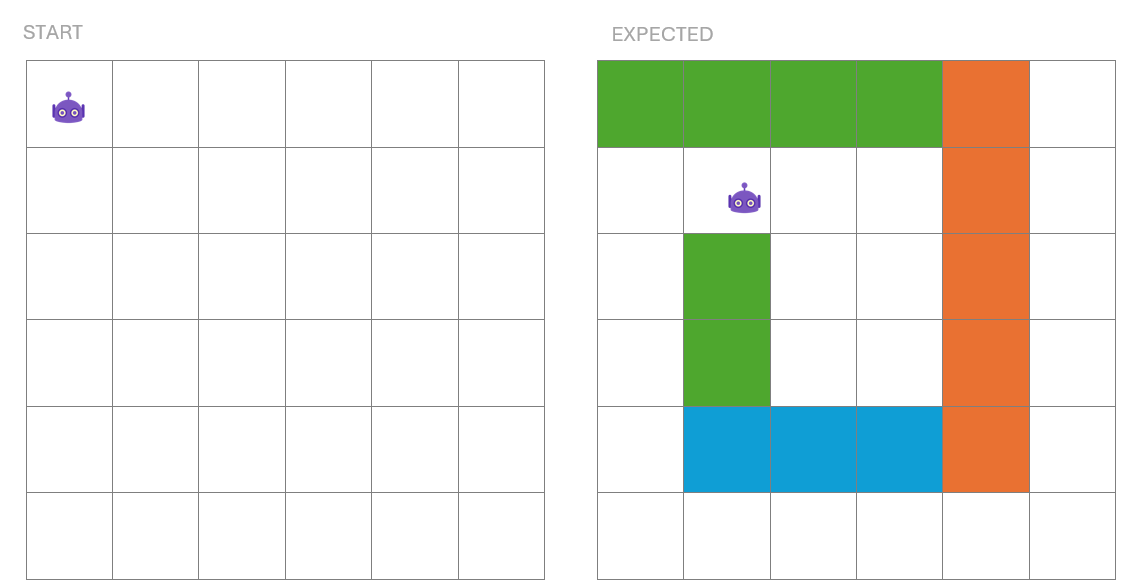
// Your code

}

**EX 4 (Draw lines in a direction (up, down, right, left)**

Now the robot needs to paint lines of **different sizes** and **colors**, and follow a specific direction (LEFT, RIGHT, UP, DOWN).

To avoid duplicated code, we use a function. *Complete the function and main to fulfill the requirements.*



Available operations:

* goRight() // Move the robot to the cell on the right
* getLeft() // Move the robot to the cell on the left
* gotUp() // Move the robot to the cell on top
* goDown() // Move the robot to the cell bellow
* paintCell(color) // paint the robot cell with given color
* RERPEAT <x> TIMES, WHILE <c>
* IF <c>, ELSE IF <c> , ELSE

MAIN

drawLine( /\* complete with parameters if needed \*/ );

drawLine( /\* complete with parameters if needed \*/ );

drawLine( /\* complete with parameters if needed \*/ );

drawLine( /\* complete with parameters if needed \*/ );

FUNCTION drawLine( /\* complete with parameters if needed \*/ ) {

// Your code

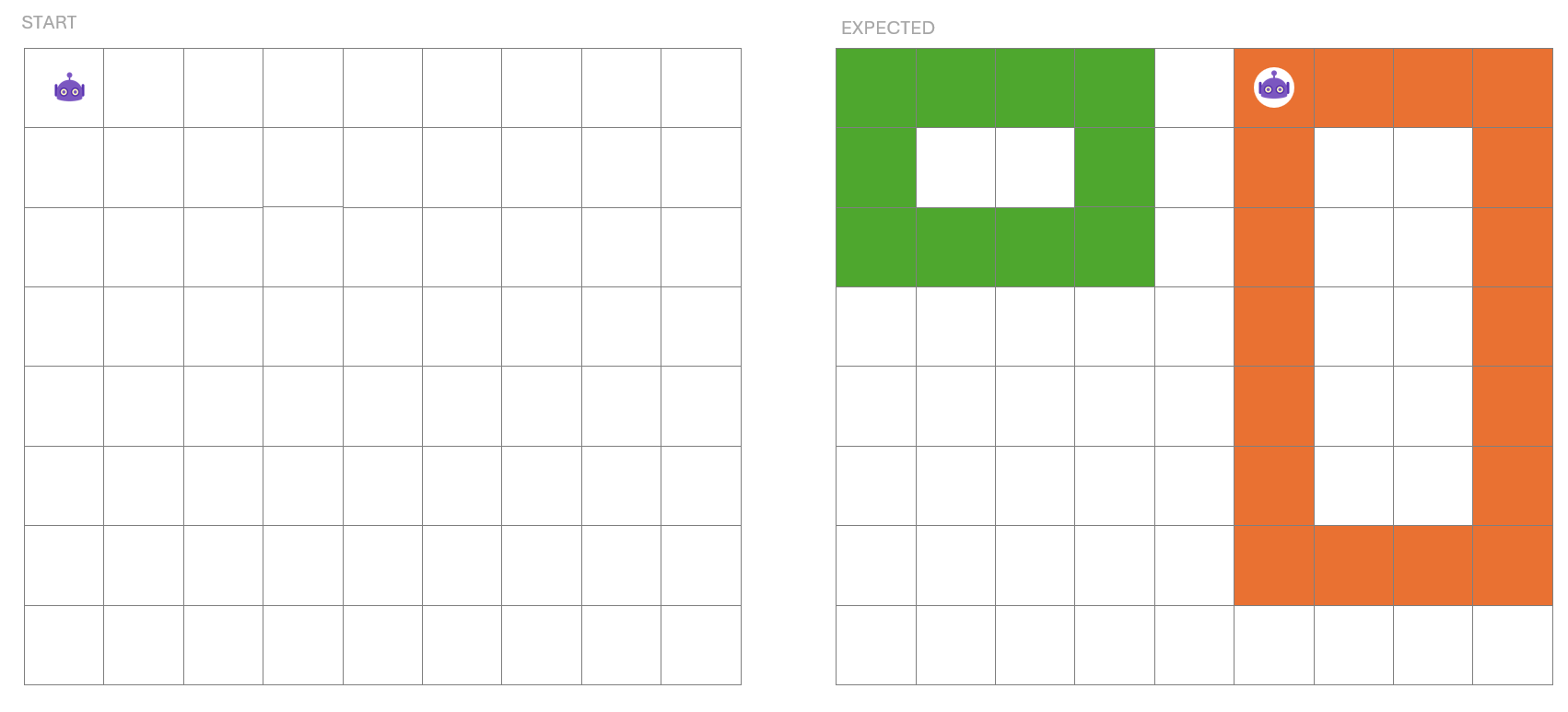
}

**EX 5 (Draw rectangles of different dimensions and colors)**

Now the robot needs to paint **rectangles** of **different dimensions** and **colors**.

**As example,** here, we paint:

* a green rectangle, dimensions 4X3
* an orange rectangle, dimensions 4X7



* To avoid duplicated code, we use a function **drawRectangle**.
* You need **to re-use the function drawLine** defined in the previous exercise

Available operations:

* drawLine(color, size, direction) // paint a line of given color, size and direction
* goRight() // Move the robot to the cell on the right
* getLeft() // Move the robot to the cell on the left
* gotUp() // Move the robot to the cell on top
* goDown() // Move the robot to the cell bellow
* RERPEAT <x> TIMES, WHILE <c>
* IF <c>, ELSE IF <c> , ELSE

MAIN

drawRectangle( /\* complete with parameters if needed \*/ );

drawRectangle ( /\* complete with parameters if needed \*/ );

FUNCTION drawRectangle ( /\* complete with parameters if needed \*/ ) {

// Your code

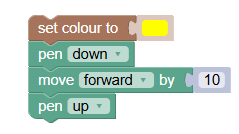
}

MANIPULATE (BLOCKY GAME)

*Time to connect to Blocky app[[1]](#footnote-2) to code and test your previous exercise!*

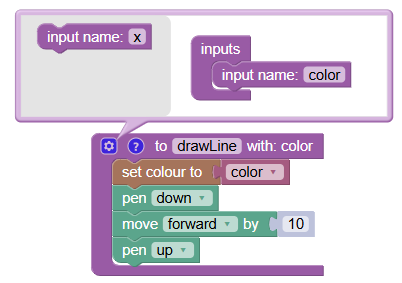
*How do I draw lines with Blocky?*

* Set the **color of the pen**
* Move the **pen down**
* **Move the turtle** anywhere, it will draw with the specified color
* Move the **pen up**



*How to create a function with Blocky?*

* **Create a function** and set its name
* **Add inputs** (parameters). *Each parameter creates a new variable*
* Write the **function body**
* **Call the function** with the requested parameters



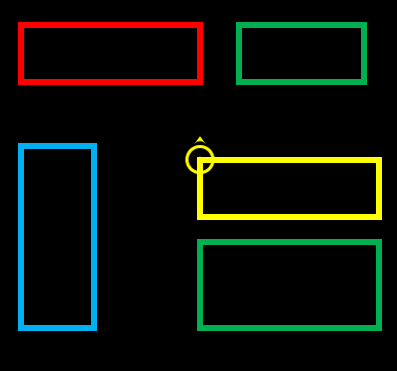
**EX 1 (Draw rectangles of different dimensions and colors.)**

Following the previous exercise, implement the 2 below functions on Blocky:

drawRectangle ( /\* complete with parameters if needed \*/ );

drawLine( /\* complete with parameters if needed \*/ );

Test the functions by drawing 5 different rectangles with the turtle.



**EX 2 (Draw your own shape - CREATIVE-MOVE!)**

Create your function(s) to **draw your own shapes**

* Your function shall take at least 3 parameters

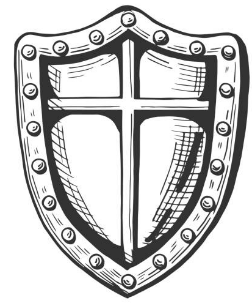
drawYourCustomShape ( /\* complete with parameters if needed \*/ );

MANIPULATE (C CODE)



*Help the knight complete his quests by writing and using C functions!*

**EX 1 (Check if the knight shield is Strong!)**



**Q1** - Implement the function **isShieldStrong** as follows:

|  |  |  |
| --- | --- | --- |
| FUNCTION NAME | isShieldStrong | |
| FUNCTION DESCRIPTION | Check whether the shield is strong or not | |
| PARAMETERS | Int | The shield strength |
| RETURN | Bool | Return **true if strength is 50** or more, otherwise false |

// Return true if strength is 50 or more, otherwise false

bool isShieldStrong(int strength) {

// TODO

}

**Q2** - Test your function in a main:

int main() {

int shield = 45;

if (isShieldStrong(shield)) {

printf("Shield is strong enough!\n");

} else {

printf("The shield might break!\n");

}

return 0;

}

**EX 2 (Find the Largest Treasure Chest !)**

**Q1** - Implement the function **max** as follows:

|  |  |  |
| --- | --- | --- |
| FUNCTION NAME | max | |
| FUNCTION DESCRIPTION | Return the max of 2 values | |
| PARAMETERS | Int | a |
|  | Int | b |
| RETURN | Int | The max between a and b |

**Q2** - Implement the function **findMaxTreasure** as follows:

WARNING : you must use the previous function max(a, b)

|  |  |  |
| --- | --- | --- |
| FUNCTION NAME | findMaxTreasure | |
| FUNCTION DESCRIPTION | Return the max of 3 treasures values | |
| PARAMETERS | Int | The value of treasure 1 |
|  | Int | The value of treasure 2 |
|  | Int | The value of treasure 3 |
| RETURN | Int | The max of the 3 treasures |

**Q3** - Test your function in a main:

int main() {

int biggest = findMaxTreasure(treasure1, treasure2, treasure3);

printf("Largest treasure: %d gold\n", biggest);

return 0;

}

**EX 3 (Analyze the Knight’s Training Stats)**



The knight trains every day for 7 days. Each day, they score a certain number of training points (from 0 to 100). The data is stored in an array.

int trainingPoints[7] = {78, 90, 45, 100, 60, 70, 85};

Your job is to help the knight analyze their weekly performance using **modular functions** !

**Q1** - Implement the bellow functions to move the knight

// Print each score on a new line.

void printAllScores(int scores[]);

// Return the highest score of the week

int getBestScore(int scores[]);

// Return the lowest score of the week

int getWorstScore(int scores[]);

// Return the average score of the week, as a double

double getAverageScore(int scores[]);

// Return true if the knight had a perfect day (100 points).

bool hadPerfectDay(int scores[]);

// Return true if the knight was able to improve himself everyday of the week

bool hasImprovedEveryDay(int scores[]);

**Q2** – Test your function using this main code:

int trainingPoints[7] = {78, 90, 45, 100, 60, 70, 85};

printAllScores(trainingPoints);

int best = getBestScore(trainingPoints);

int worst = getWorstScore(trainingPoints);

double avg = getAverageScore(trainingPoints);

bool perfect = hadPerfectDay(trainingPoints);

bool improved = improvedDuringWeek(trainingPoints);

printf("\nBest Score: %d\n", best);

printf("Worst Score: %d\n", worst);

printf("Average Score: %.2f\n", avg);

printf("Had Perfect Day: %s\n", perfect ? "true" : "false");

printf("Improved During Week: %s\n", improved ? "true" : "false");

The above code should provide the bellow output:

All Scores:

78

90

45

100

60

70

85

Best Score: 100

Worst Score: 45

Average Score: 75.43

Had Perfect Day: true

Improved During Week: true

1. https://blockly.games/turtle?lang=en&level=10 [↑](#footnote-ref-2)