

TEACHERS' GUIDE

Table of Contents

Organizing you work with Scottie Go! — 7

Notes for teachers — 9

Module I - Europe 1 — 11

Module Introduction - commands — 12

Quest 1 - controlling the game character, first commands — 13

Quest 2 — 15

Quest 3 - digits, creating numbers, learning addition and subtraction — 16

Quest 4 - making a turn — 18

Quest 5 — 20

Quest 6 — 21

Quest 7 — 23

Quest 8 - walking backwards — 25

Quest 9 — 27

Quest 10 — 29

Module II - Europe 2 — 31

Module introduction: picking up objects from the game-board — 32

Quest 1 — 33

Quest 2 — 35

Quest 3 — 37

Quest 4 — 39

Quest 5 - jumping — 41

Quest 6 — 44

Quest 7 - note — 46

Quest 8 - note — 49

Quest 9 — 52

Quest 10 — 55

Table of Contents

Module III - South America — 57

Module Introduction - the REPEAT loop — 58

Quest 1 - the first loop — 59

Quest 2 — 62

Quest 3 - working inside a loop and outside of a loop — 64

Quest 4 - Scottie GO! EDU as a tool for introducing notions in maths and physics — 67

Quest 5 — 70

Quest 6 — 72

Quest 7 - nested loops — 74

Quest 8 — 77

Module IV - North America — 79

Quest 1 - beaver as a bridge — 80

Quest 2 — 82

Quest 3 — 84

Quest 4 - beaver in motion — 86

Quest 5 — 89

Quest 6 - using REPEAT loop and the SWITCH TO tiles together — 92

Quest 7 - the Beaver makes a turn — 95

Quest 8 — 98

Quest 9 — 100

Quest 10 — 102

Module V - Antarctic — 104

Module Introduction - interaction with Scottie's surroundings - ACTIVATE and PLACE tiles — 105

Quest 1 - the ACTIVATE tile — 107

Quest 2 - the PLACE tile — 110

Quest 3 — 113

Table of Contents

Module V - Antarctic ctnd. — 104

...

Quest 4 — 115

Quest 5 — 117

Quest 6 — 120

Quest 7 - SWITCH TO the beavers and move ice floes — 122

Quest 8 — 125

Quest 9 — 127

Quest 10 — 130

Module VI - Asia — 132

MODULE INTRODUCTION - conditional loop - REPEAT WHILE — 133

Quest 1 - using the REPEAT WHILE tile — 135

Quest 2 — 138

Quest 3 — 141

Quest 4 - REPEAT WHILE with other conditions — 144

Quest 5 - REPEAT WHILE nested in a loop — 147

Quest 6 — 150

Quest 7 — 153

Quest 8 — 155

Module VII - Australia 1 — 158

MODULE INTRODUCTION - new concept: VARIABLES — 159

Quest 1 - INTRODUCTION - building aphalt roads — 162

Quest 2 — 165

Quest 3 — 167

Quest 4 - X and Y — 170

Quest 5 — 173

...

Table of Contents

Module VII - Australia 1 ctnd. — 158

...

Quest 6 - variables and loops — 175

Quest 7 — 178

Quest 8 - mathematical operations on variables — 180

Quest 9 — 183

Quest 10 — 185

Module VIII - Australia 2 — 187

Module introduction - conditional statements - IF, ELSE — 188

Quest 1 - action square — 192

Quest 2 — 195

Quest 3 — 197

Quest 4 — 199

Quest 5 — 201

Quest 6 - using multiple conditional statements — 204

Quest 7 — 207

Quest 8 - ELEVATED SQUARE and HERE tiles — 209

Module IX - North America 2 — 212

Module introduction (A) - ELSE IF — 213

Module introduction (B) - the ELSE tile — 216

Module introduction (C) - [optional] - REPEAT (FOREVER) — 219

Quest 1 - ELSE — 222

Quest 2 - ELSE IF and MARKS — 226

Quest 3 — 231

Quest 4 — 233

Quest 5 — 235

...

Table of Contents

Module IX - North America 2 ctnd. — 212

...

Quest 6 — 238

Quest 7 - conveyor belts — 240

Quest 8 - switches — 243

Quest 9 — 246

Module X - Africa — 248

Module introduction - functions — 249

Quest 1 - Pierwsza funkcja, WZNIESIENIA — 253

Quest 2 — 257

Quest 3 — 259

Quest 4 - Drawing — 261

Quest 5 — 264

Quest 6 - Using a tile multiple times — 266

Quest 7 — 269

Quest 8 - Managing functions — 271

Organizing your work with Scottie Go! (1/2)

Make sure the workplace is well-lit

Playing with Scottie Go! requires your pupils to take photos of the code they wrote. Not enough lighting may make it impossible to scan the code properly.

Never combine tiles from different kits

Each tile in a kit has its unique ID number and graphics recognized by the camera of your mobile device. If you mix up tiles from several sets in one code, they may not be recognized properly. We recommend that you make a mark on the back of tiles from each kit with a separate colour to distinguish them from tiles that belong to other kits.

NEVER COMBINE THE HOME VERSION WITH THE EDUCATIONAL VERSION TILES

In the optimal model we recommend to use one kit box per 2 pupils

Before you BEGIN using Scottie Go! EDU with your pupils, make sure they know the rules, e.g. you may want to make a contract with your pupils. If you do so, make sure that one of the rules tells your pupils not to lend each other tiles from their kits. When the class finishes, make sure they pack the tiles back into their respective boxes so as not to mix up the tiles from different kits. If you work with younger children, we recommend that you prepare all the required tiles in advance.

Create a special teacher account that gives you access to all of the quests

You can create a teacher account that will give you access to all quests in all modules. You will be able to access any quest even if you haven't completed the whole Scottie Go! course before. The options accessible through the teacher account allow you to use Scottie Go! while teaching various subjects to cover the programming concepts of your choice, depending on your pupils' age and skill level.

To create the **Teacher** account, choose the following option in the **Scottie Go! Edu** app.

SELECT PLAYER -> NEW PLAYER and then enter this code:

Teacher_13_cps#

DO NOT REVEAL THIS CODE TO PUPILS!

Organizing your work with Scottie Go! (2/2)

The Teacher account allows your pupils to work in teams. In this model you will need only one device to display the Quest description and one mobile device to scan the programs created by your pupils.

The Teacher account allows you to make the best use of Scottie Go! progress cards. To download additional materials, log into your account at:
www.scottiego.com/education

Notes for teachers (1/2)

There is more than one correct solution

This guide provides you with solutions to each quest. Remember that there are numerous ways to solve the same problems with various programs - some of them may even be shorter than the suggested solution.

The game comprises 179 tiles that are active in every module and in every quest

You can solve each quest again and again, using the new programming concepts as well as new tiles introduced in subsequent quests and modules.

Scottie Go! Edu app employs the mechanics of a game, but is a full-fledged educational tool

While using Scottie GO! Edu app, you will be able to both meet your teaching curriculum goals in terms of key coding competencies and tap into the latest world developments in engaging computational thinking in other contexts:

• understanding, analysing and solving problems based on logical and abstract thinking, computational thinking and visual presentation of data,

• programming and problem-solving with the use of computers and other digital devices: assembling and programming algorithms, organising, searching for and sharing information, using computer applications,

• developing social competences, such as: group communication and collaboration, including virtual environments, participation in group projects and project management,

REMEMBER - OUR GAME IS JUST A TOOL THAT ASSISTS YOUR PUPILS IN LEARNING, NOT A GOAL IN ITSELF!

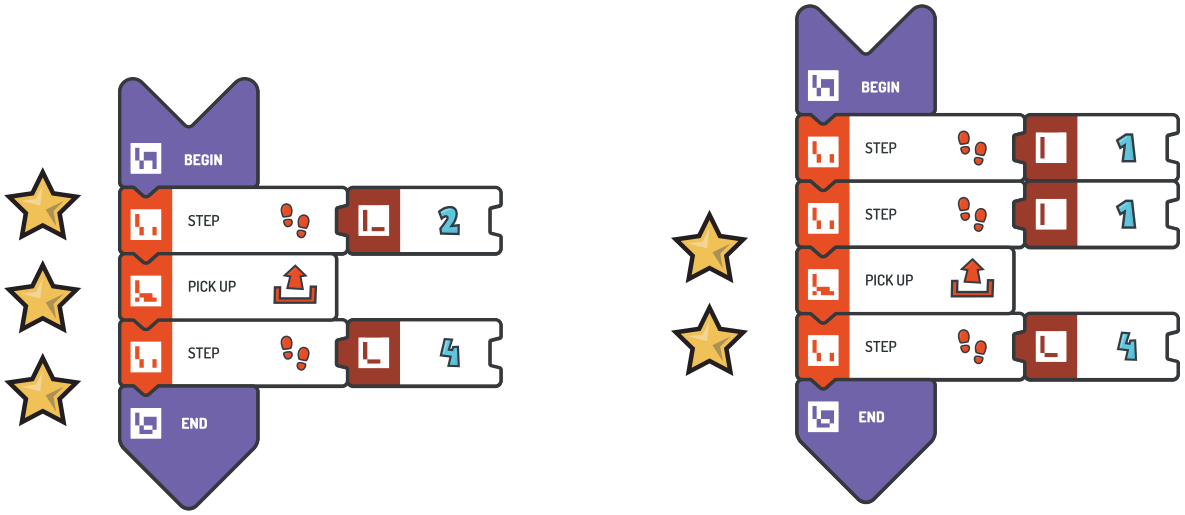
While using Scottie GO! Edu during your lessons, we recommend that you follow the rules related to active methods, lesson objectives, lesson management as well as evaluation. Scottie Go! EDU follows the spiral approach to the newly introduced notions, i.e. in one quest introduces a command which is also used in the subsequent quests in different context. The difficulty level of quests increases with each quest as the game progresses, which allows the teacher and his/her pupils to develop their algorythmic skills. Each time a new command is introduced, it is signalled by an exclamation mark in the top right-hand corner of the game character on your screen.

Notes for teachers (2/2)

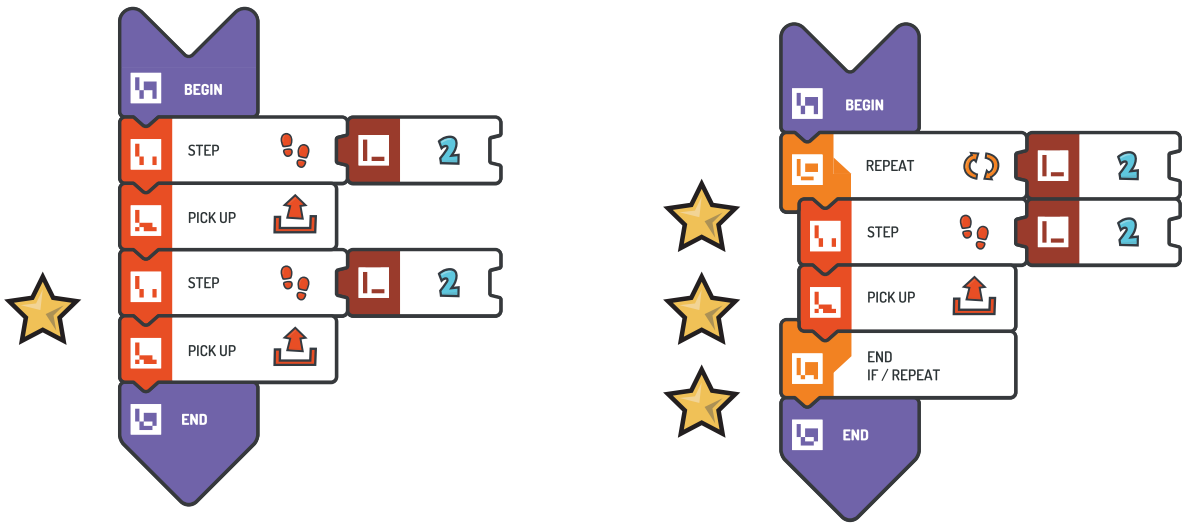
A good coder and an ambitious pupil should always look for an optimal way to solve a problem. The program is "optimal" when it is constructed using as few lines of code as possible and it still achieves its objective.

During their lessons teachers should create the atmosphere that facilitates learning. The **ScottieGo! Edu** app features some elements of formative assessment in an attractive game-based environment. After completing each quest, your pupils will receive from 1 to 3 stars as a reward for the quality and optimisation of their codes. It's a lot of fun without making them compete against each other.

Three stars if the program is the best, two stars if the solution is a bit less efficient. One star means that the quest was solved, but the program was too long.



In some quests, if players want to be granted more than one star, they will need to use specific tiles, e.g. the **REPEAT** tile, in their solution.



Module X
Africa



Defining functions.

Scottie wants to finish repairing his spaceship and complete the assignements he promised to his animal friends. Plant the missing seedlings in the park, draw lines to mark where a path should be. Don't forget to use functions while you are at that!

MODULE INTRODUCTION 1/4: Functions

Usually, when writing a program, we use the same sets of commands time and time again. We soon realise that arranging them multiple times is a waste of time and it's better to find a solution similar to the one used in case of variables - let's put a set of commands into special containers and call them out when we need them. This will make our program simpler, shorter and clearer.

Such containers are called **functions** (subprograms) and in order to use them, you are going to need to define and call them in the app using specific sets of tiles.

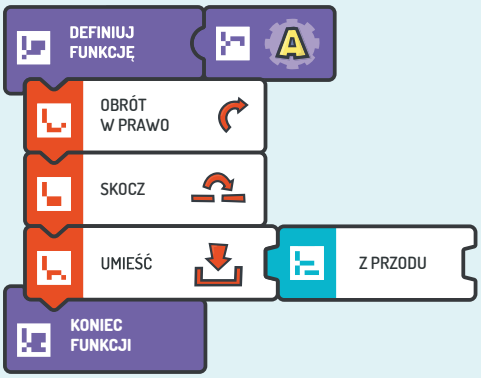
To create a function, we need to use the **DEFINE FUNCTION** tile and assign this function a **name: A, B or C** (in Scottie Go! we can define up to three functions at a time):



We will also need the **END FUNCTION** tile.



Between the **DEFINE FUNCTION** and **END FUNCTION** tiles we should place the sequence of commands that we will need to use in the program repeatedly, e.g.:



Once we have scanned a function, we can use it in the program by using the **CALL FUNCTION** tile with the **function name (A, B or C)**. In this case:



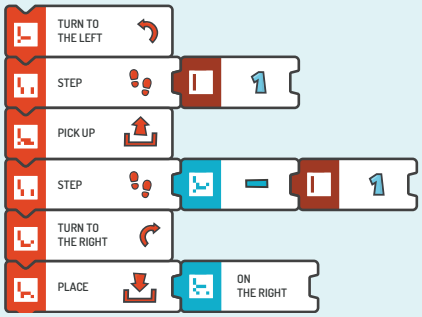
MODULE INTRODUCTION 2/4: Functions

Example:

In the quest presented below **Scottie is supposed to pick and plant seedlings** in designated places (where the soil was prepared for them). **As you can see, each of those tasks must be performed twice.** Let's revise creating and using functions on this example.

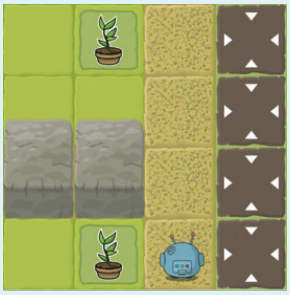
1. First **arrange a sequece of instructions** responsible for collecting and planting a seedling in a designated spot.

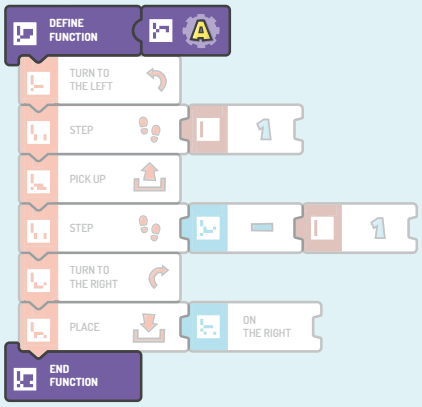




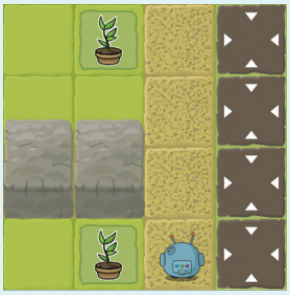
2. Then use the previously assembled set of instructions to **define a function.**

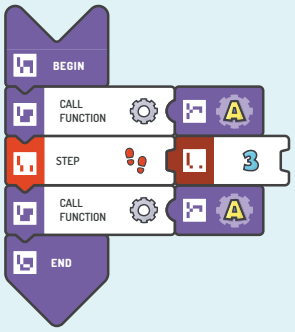
Let's name it A, B or C.





3. Build a program in which you can call our function by name when needed.





MODULE INTRODUCTION 3/4: Functions [extra]

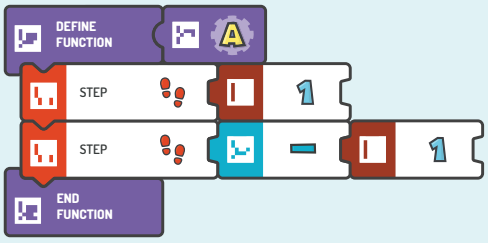
We can place other functions inside a function.

By using functions, we can build a very complex programs in which:
1. First and foremost, we will use fewer tiles (thanks to functions your code will be shorter and easier to manage).
2. Second, we will be able to better focus on the tasks that Scottie should perform. **By introducing the notion of functions, we raise the level of abstraction, which happens to us every day when we learn such tasks as tying our shoes.**

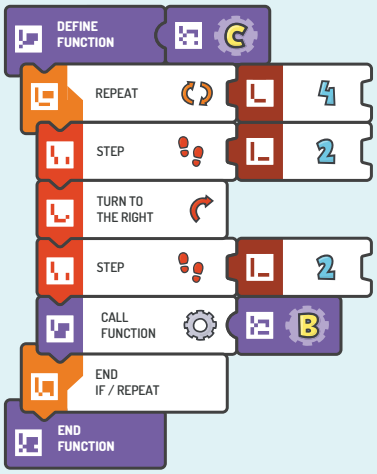
For example, when we consider the process of learning how to „tie our shoes", we perceive it as a single task, even though it comprises a complex set of movements that involve manipulating shoelaces in a precise order.

What path will Scottie take when you:

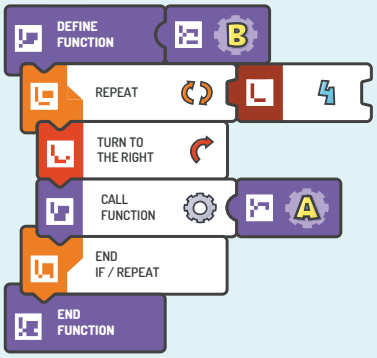
Call function A?



Call function C?



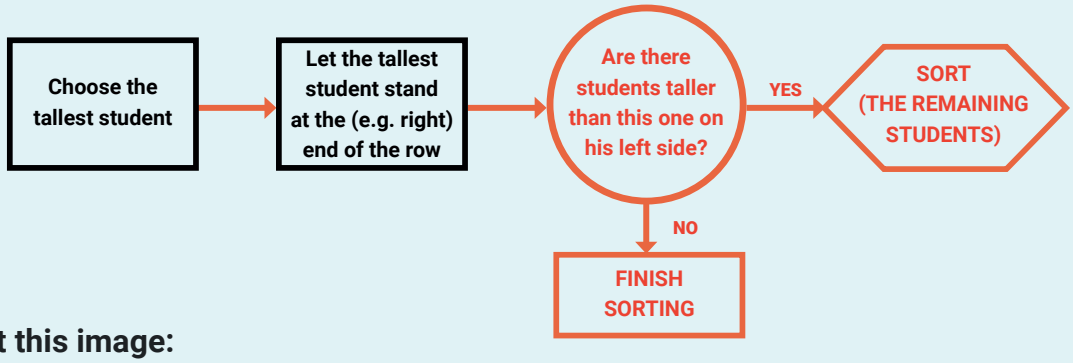
Call function B?



MODULE INTRODUCTION 4/4: Functions [extra]

A function can call itself!

We call such a **recursive function**. Recursive functions are used for example in sorting algorithms, i.e. algorithms that are used to put a set of values in order according to their size or other key (e.g. when we want to sort books by title). Let's look at the algorithm that can be used to sort students by their height::



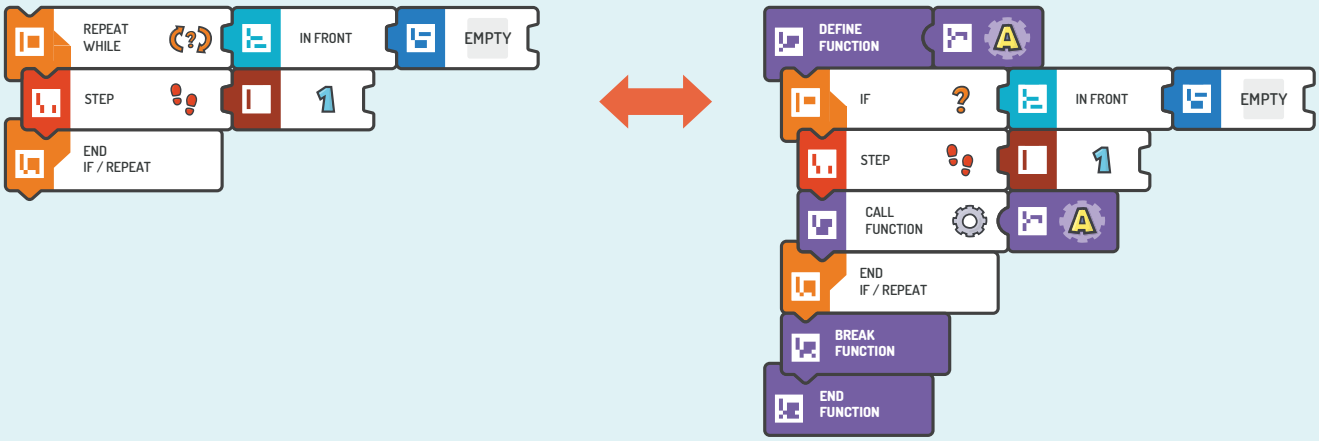
or **look at this image:**



The key element in recursive functions is the conditional expression that will break the function if the condition is true. Without it, the function will never be broken and will generate an error (or will block the computer until the memory is out). In order to stop the function, we need the **BREAK FUNCTION** tile.



Using a recursive function, we can create our own conditional loop:



INTRODUCTION:
First function, ELEVATED SQUARES (1/2)

In this quest you are supposed to call your custom function A by using the **CALL FUNCTION** tile together with the **A** tile (where A stands for the function name).



But first you need to define this function by creating a sequence of commands to be performed when the function is called. Use the **DEFINE FUNCTION** tile together with the **A** tile (where A stands for the function name)



and the **END FUNCTION** tile placed at the end.



For example:



In the text format:

```
DEFINE FUNCTION A {
        STEP (2)
}
```

REMEMBER!

Step 1 - RECORD FUNCTIONS (up to three functions: A, B, C)

Step 2 - SCAN FUNCTIONS (up to three functions: A, B, C)

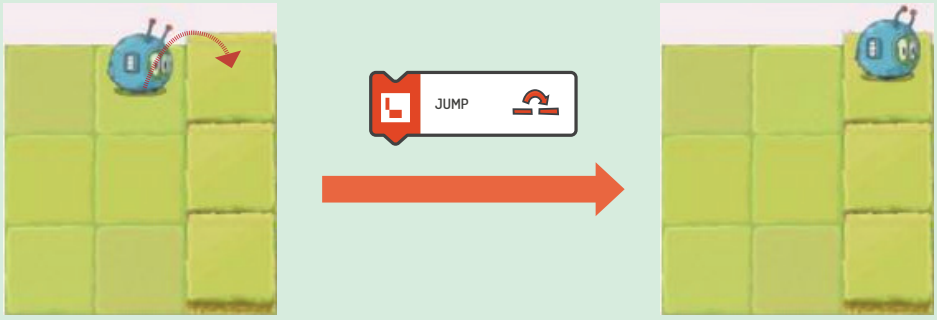
Step 3 - WRITE A PROGRAM IN WHICH THE FUNCTIONS ARE CALLED

Step 4 - SCAN AND RUN THE PROGRAM

Learn more about functions in the **introduction to Module X**.

INTRODUCTION:
First function, ELEVATED SQUARES (2/2)

In some of the quests in this module you might want to make Scottie jump onto elevated squares in order to solve the quest with fewest lines possilbe.

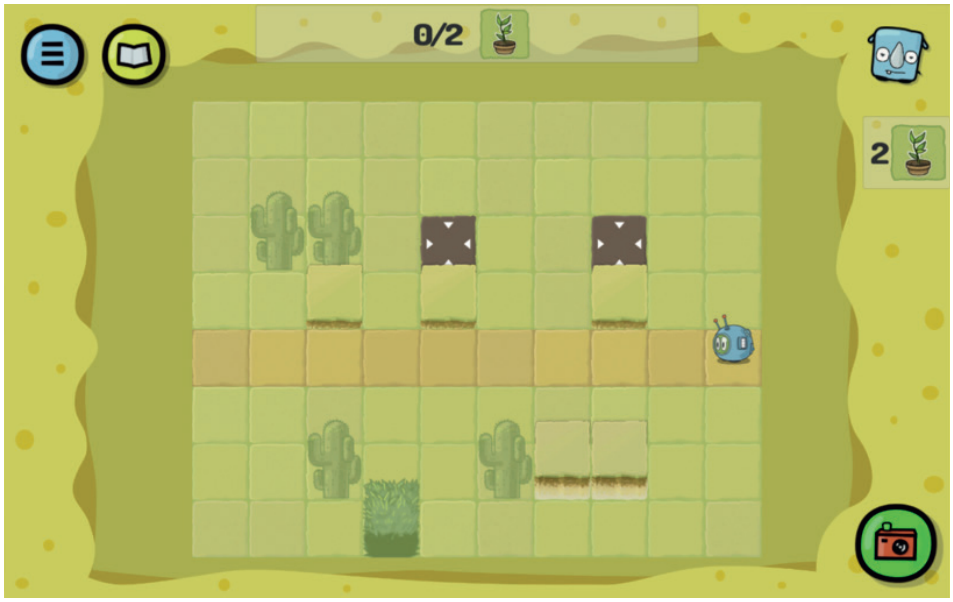


The only way to reach an elevated square is to use the **JUMP** tile. To get down, you can use either the **JUMP** or the **STEP** tile.

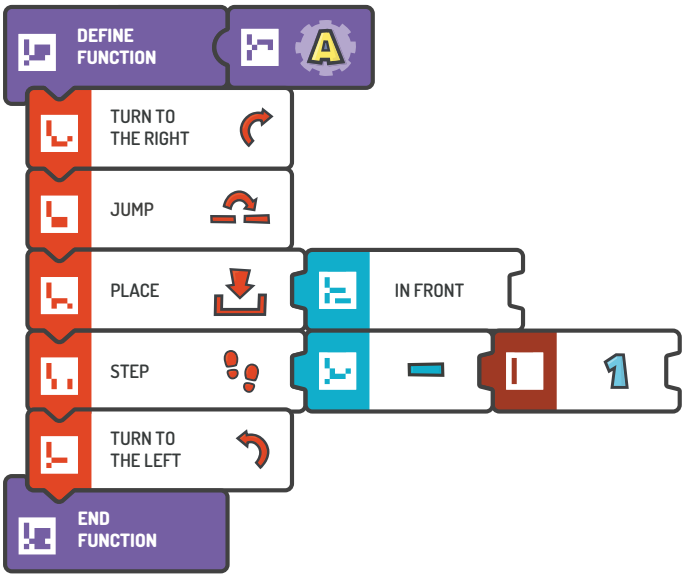
You can also refer to elevated squares in your code by using the **ELEVATED SQUARE** tile.

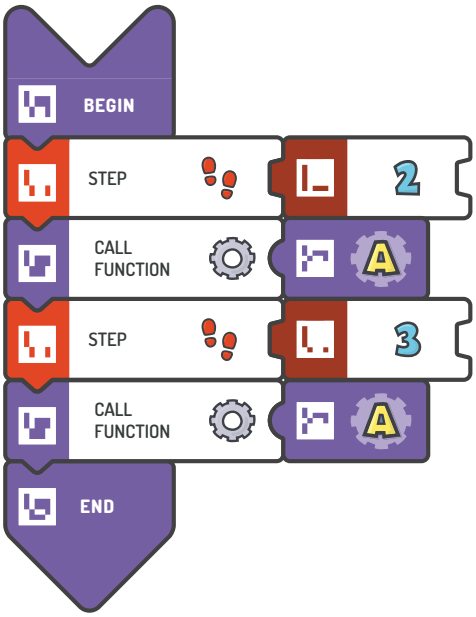


Challenge **(Build and call your own, custom function, plant seedlings on action squares)**:

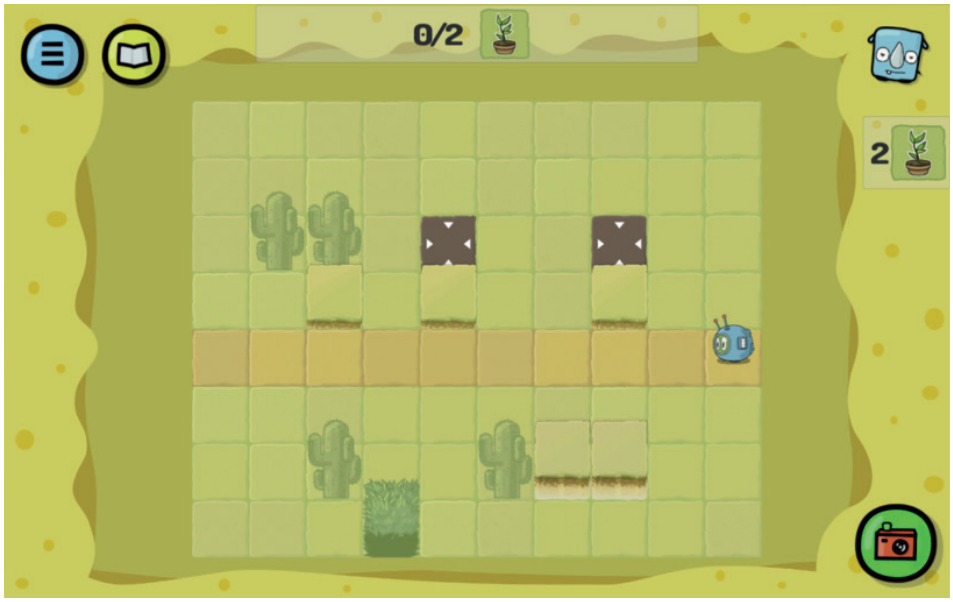


Example solution:





Challenge **(Build and call your own, custom function, plant seedlings on action squares)**:



Example solution (in text format):

```
DEFINE FUNCTION A {
        TURN TO THE RIGHT
        JUMP
        PLACE (IN FRONT)
        STEP (-1)
        TURN TO THE LEFT
}

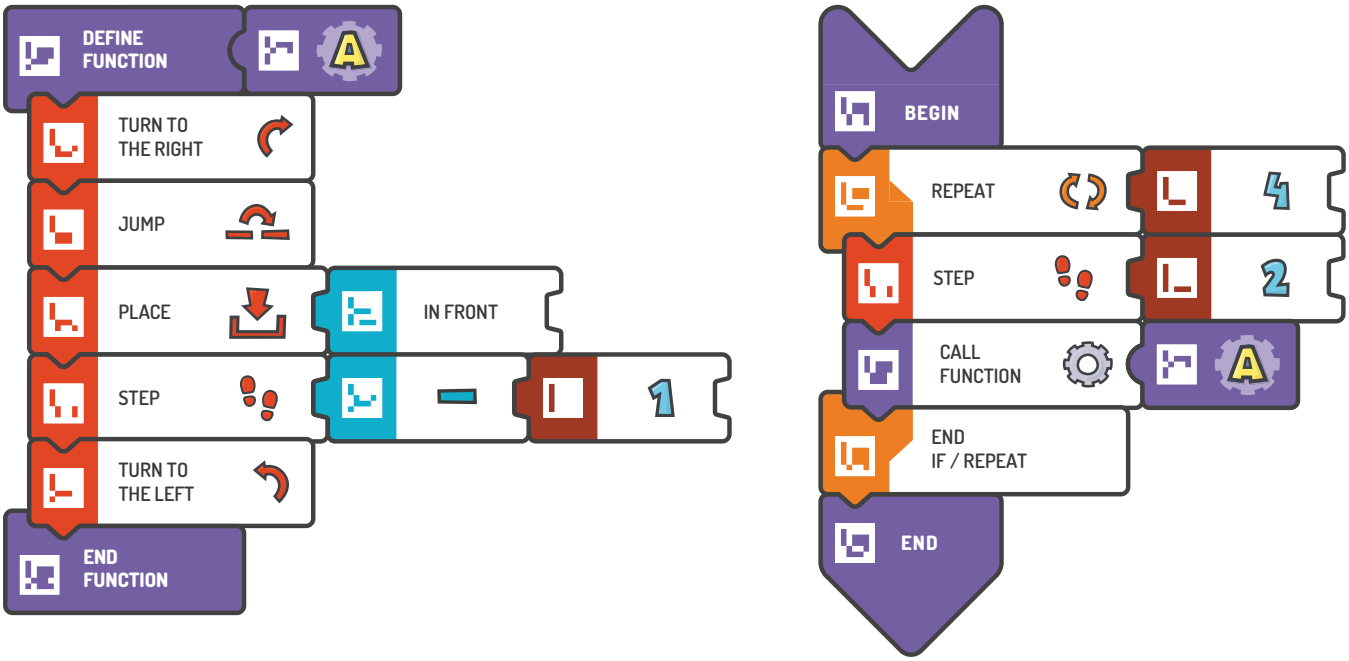
BEGIN
        STEP (2)
        CALL FUNCTION A
        STEP (3)
        CALL FUNCTION A
END
```

Challenge

(Call your own custom function four times using the repeat loop):



Example solution:



Challenge

(Call your own custom function four times using the repeat loop):



Example solution (in text format):

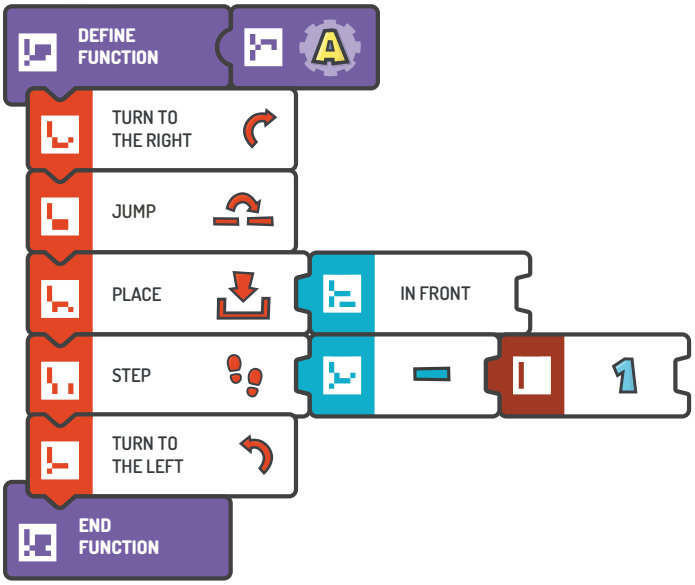
```
DEFINE FUNCTION A {
        TURN TO THE RIGHT
        JUMP
        PLACE (IN FRONT)
        STEP (-1)
        TURN TO THE LEFT
}

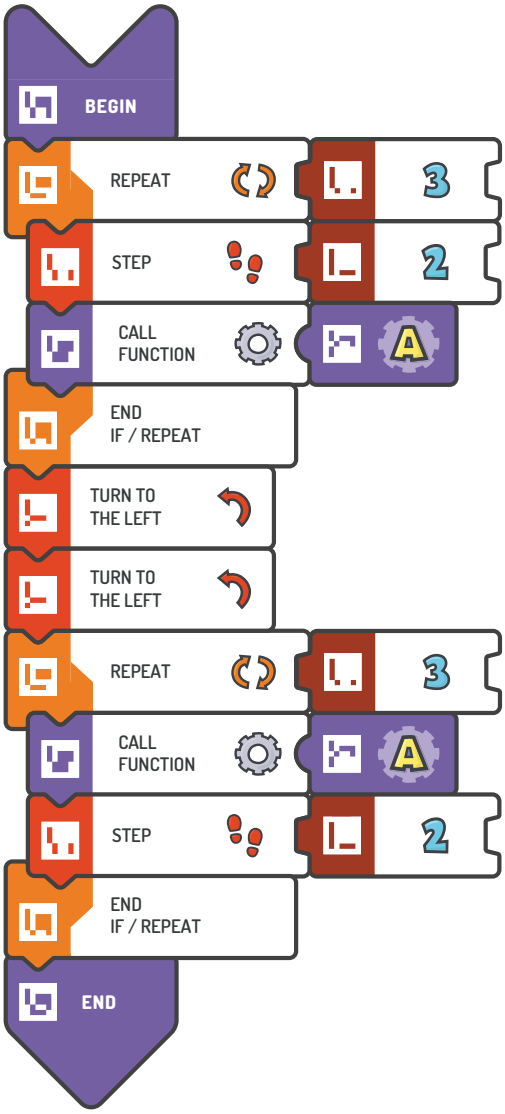
BEGIN
        REPEAT (4) {
                STEP (2)
                CALL FUNCTION A
        }
END
```

Challenge:



Example solution:





Challenge:



Example solution (in text format):

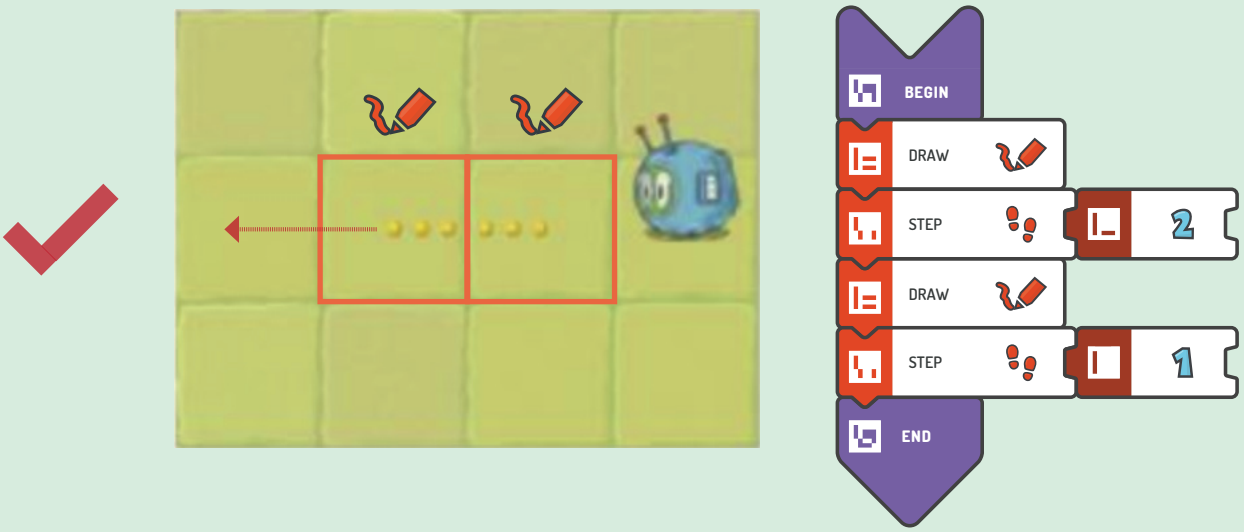
```
DEFINE FUNCTION A {
        TURN TO THE RIGHT
        JUMP
        PLACE (IN FRONT)
        STEP (-1)
        TURN TO THE LEFT
}

BEGIN
        REPEAT (3) {
                STEP (2)
                CALL FUNCTION A
        }
        TURN TO THE LEFT
        TURN TO THE LEFT
        REPEAT (3) {
                CALL FUNCTION A
                STEP (2)
        }
END
```

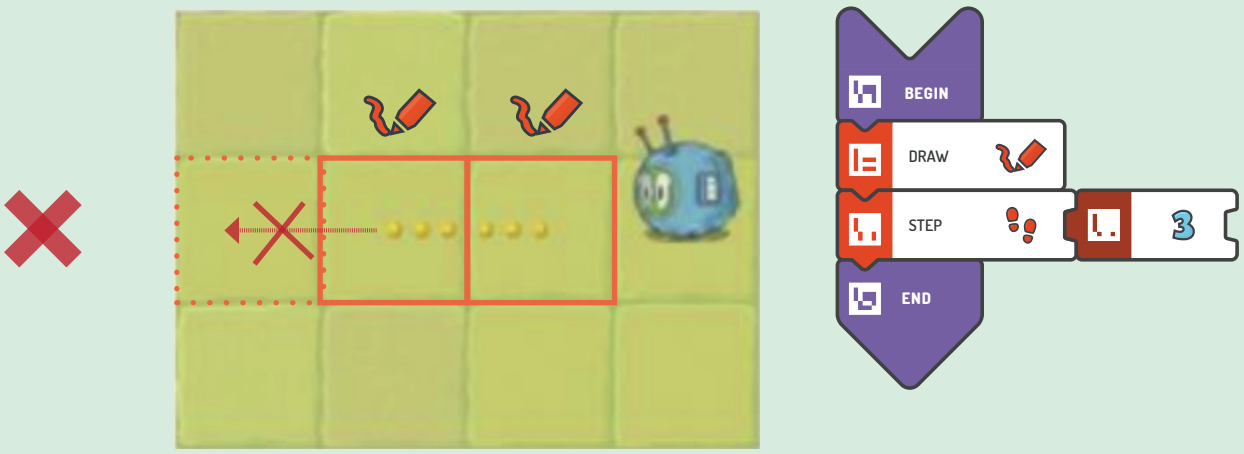
INTRODUCTION:
Drawing

Scottie was assigned a task to draw lines marking the route for the paths which are planned to be built in the park.

In order to complete this task, we will use the **DRAW** tile. **Use this tile both to start and to finish drawing.** When in the drawing mode, **Scottie will leave a line on each square he enters or goes through.**



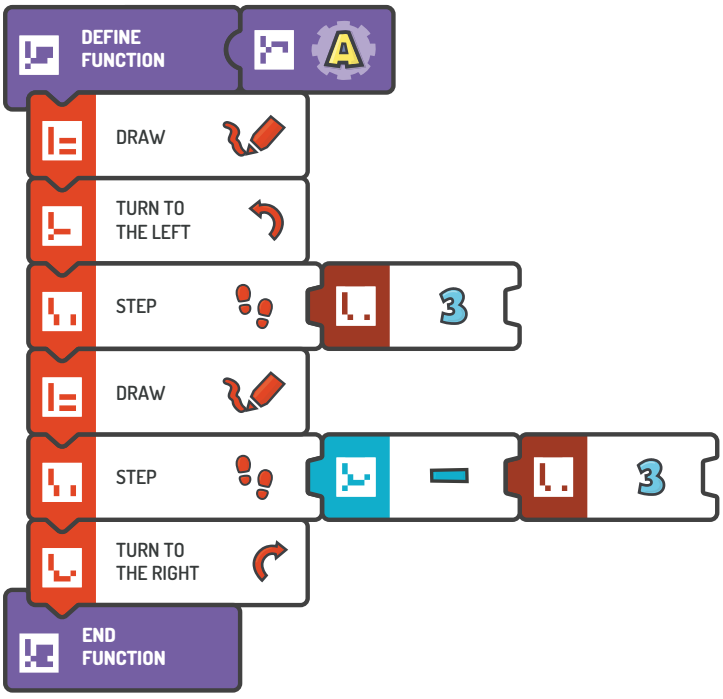
Drawing on squares without dots will be considered as an error.

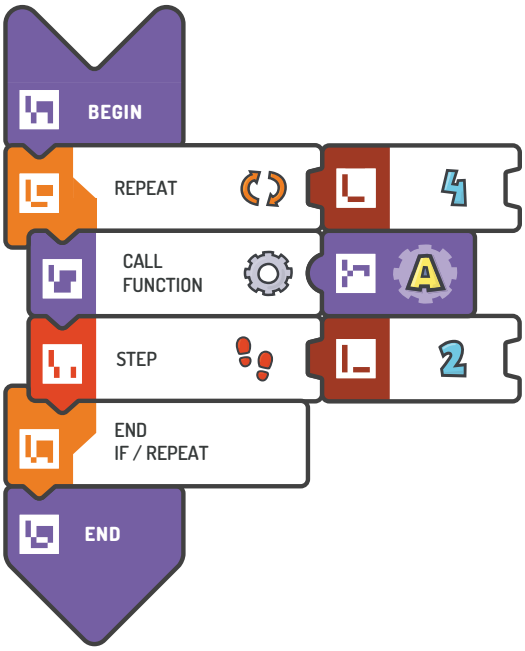


Challenge **(Create your own custom function A and place the DRAW tile inside it)**:



Example solution:





Challenge **(Create your own custom function A and place the DRAW tile inside it)**:



Example solution (in text format):

```
DEFINE FUNCTION A {
        DRAW
        TURN TO THE LEFT
        STEP (3)
        DRAW
        STEP (-3)
        TURN TO THE RIGHT
}

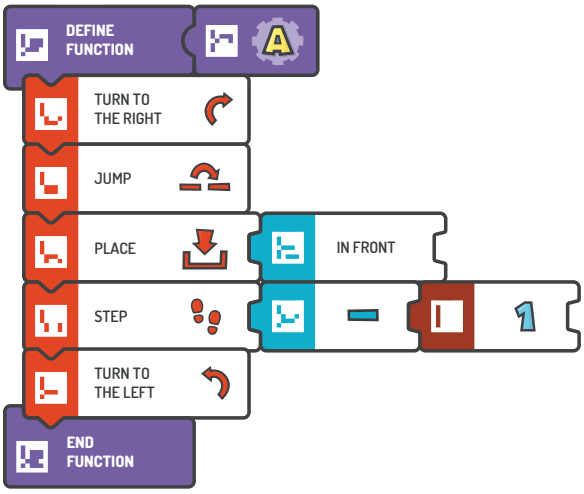
BEGIN
        REPEAT (4) {
                CALL FUNCTION A
                STEP (2)
        }
END
```

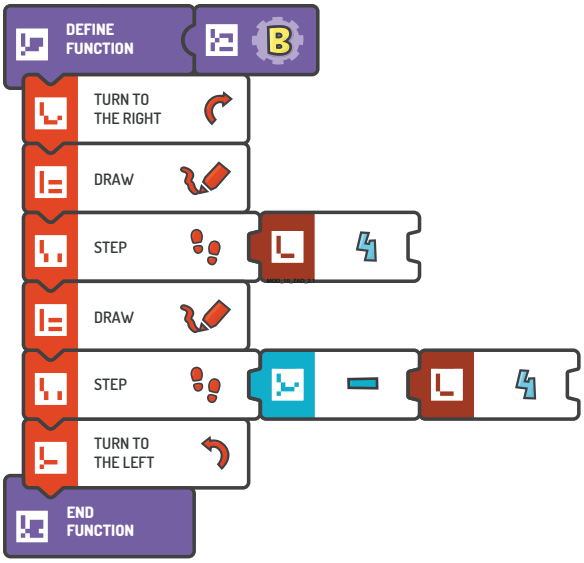
Challenge

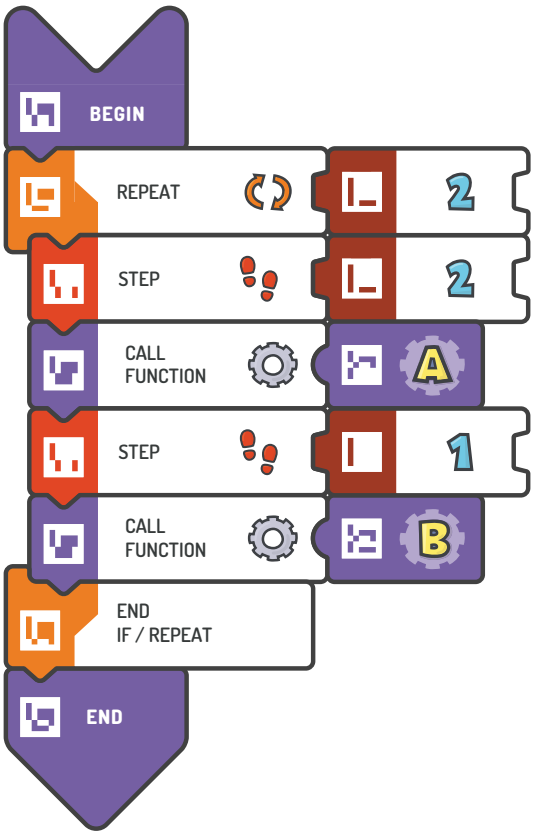
(Create your two custom functions A and B and then use them in the code):



Example solution







Challenge

(Create your two custom functions A and B and then use them in the code):



Example solution (in text format):

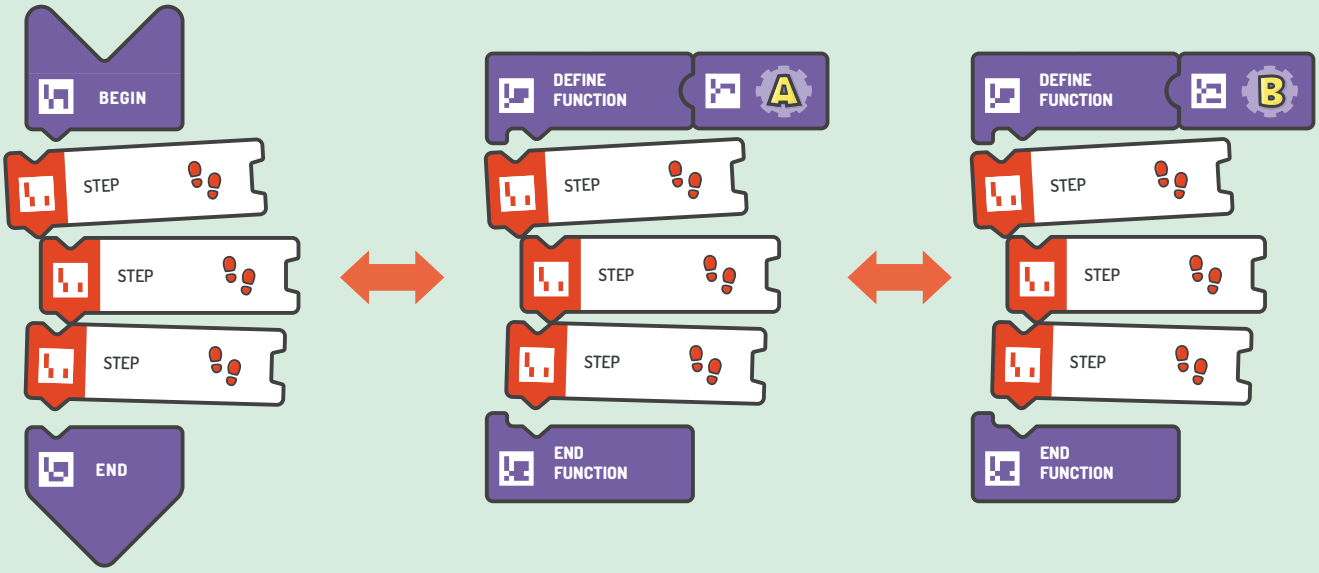
```
DEFINE FUNCTION A {
        TURN TO THE RIGHT
        JUMP
        PLACE (IN FRONT)
        STEP (-1)
        TURN TO THE LEFT
}

DEFINE FUNCTION B {
        TURN TO THE RIGHT
        DRAW
        STEP (4)
        DRAW
        STEP (-4)
        TURN TO THE LEFT
}
```

```
BEGIN
        REPEAT (2) {
                STEP (2)
                CALL FUNCTION A
                STEP (1)
                CALL FUNCTION B
        }
END
```

INTRODUCTION
Using a tile multiple times

The number of the tiles in the set is limited, but you can use the same tiles when defining different functions and in the program itself, **i.e. you can switch tiles between functions and the program when you scan them.**

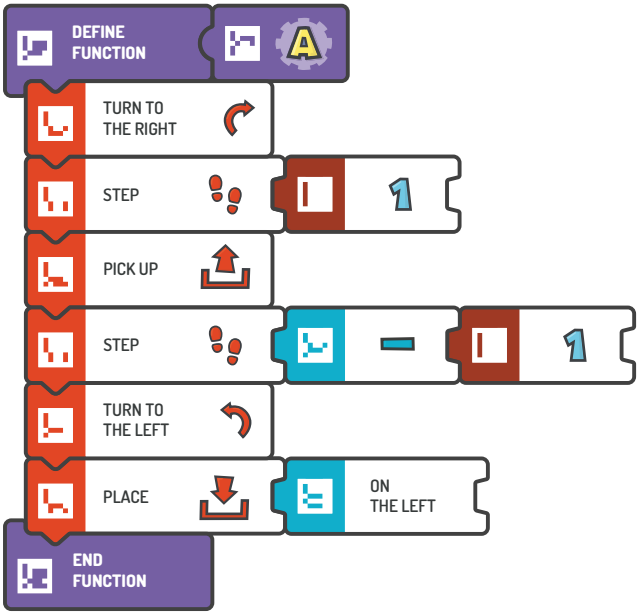


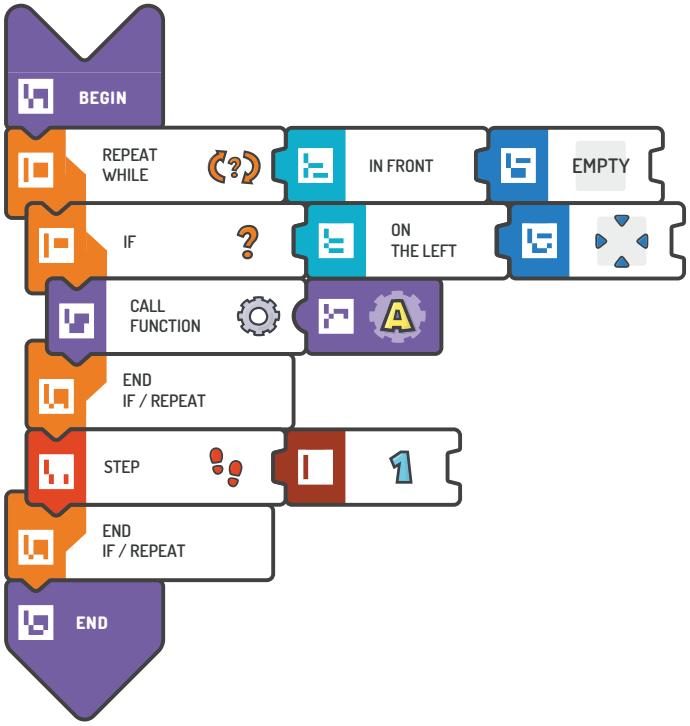
Challenge

(Use the REPEAT WHILE and the IF tiles in this quest):



Example solution:





Challenge

(Use the REPEAT WHILE and the IF tiles in this quest):



Example solution (in text format):

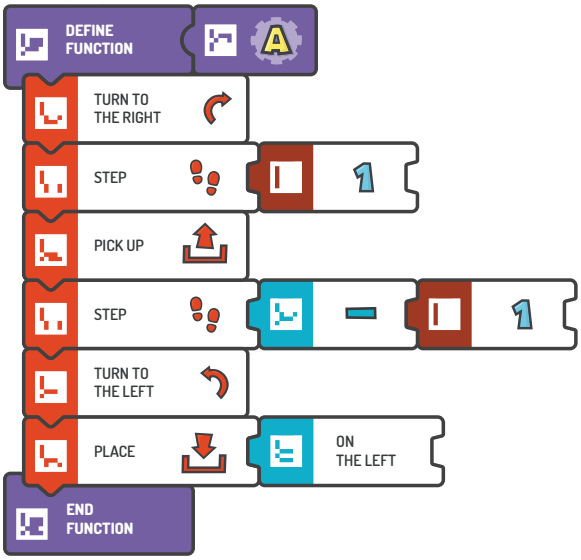
```
DEFINE FUNCTION A {
        TURN TO THE RIGHT
        STEP (1)
        PICK UP
        STEP (-1)
        TURN TO THE LEFT
        PLACE (ON THE LEFT)
}

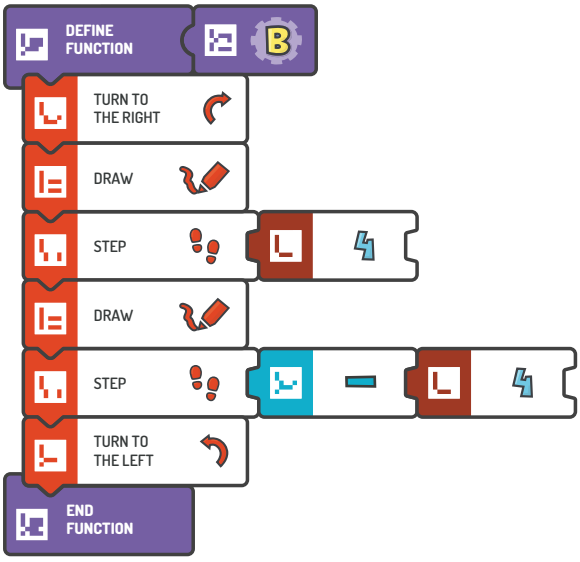
BEGIN
        REPEAT WHILE (IN FRONT < EMPTY) {
                IF (ON THE LEFT < ACTION SQUARE) {
                        CALL FUNCTION A
                }
                STEP (1)
        }
END
```

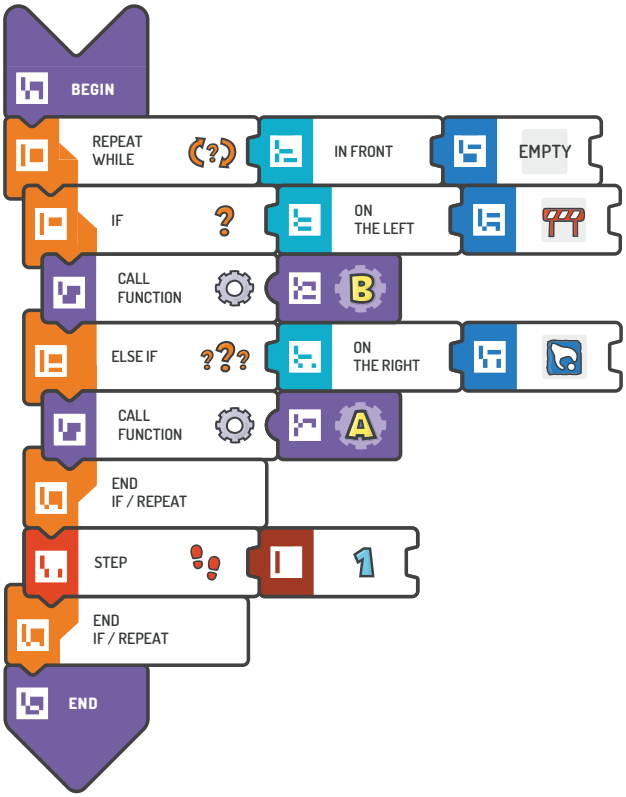
Challenge:



Example solution:







Challenge:



Example solution (in text format):

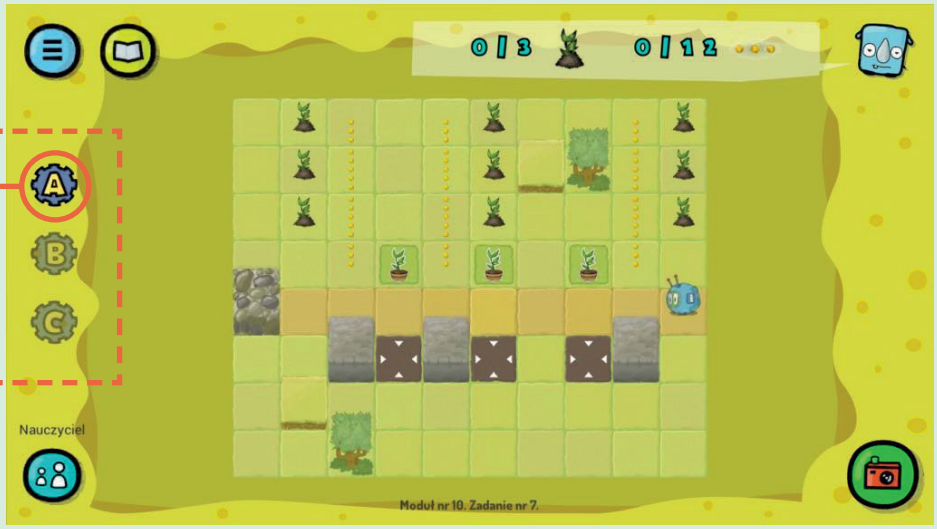
```
DEFINE FUNCTION A {
        TURN TO THE RIGHT
        STEP (1)
        PICK UP
        STEP (-1)
        TURN TO THE LEFT
        PLACE (ON THE LEFT)
}

DEFINE FUNCTION B {
        TURN TO THE RIGHT
        DRAW
        STEP (4)
        DRAW
        STEP (-4)
        TURN TO THE LEFT
}

BEGIN
        REPEAT WHILE (IN FRONT < EMPTY) {
                IF (ON THE LEFT < OBSTACLE) {
                        CALL FUNCTION B
                } ELSE IF
                (ON THE RIGHT < COLLECTABLE OBJECT) {
                        CALL FUNCTION A
                }
                STEP (1)
        }
END
```

INTRODUCTION:
Managing functions

In Scottie Go! we can define up to three functions (**A, B, C**) to be used in one program. If you scan a function, its icon will appear along the left edge of the screen.



Functions that have not been defined yet are greyed-out. Tap the name of the defined function to display its definition on the screen.

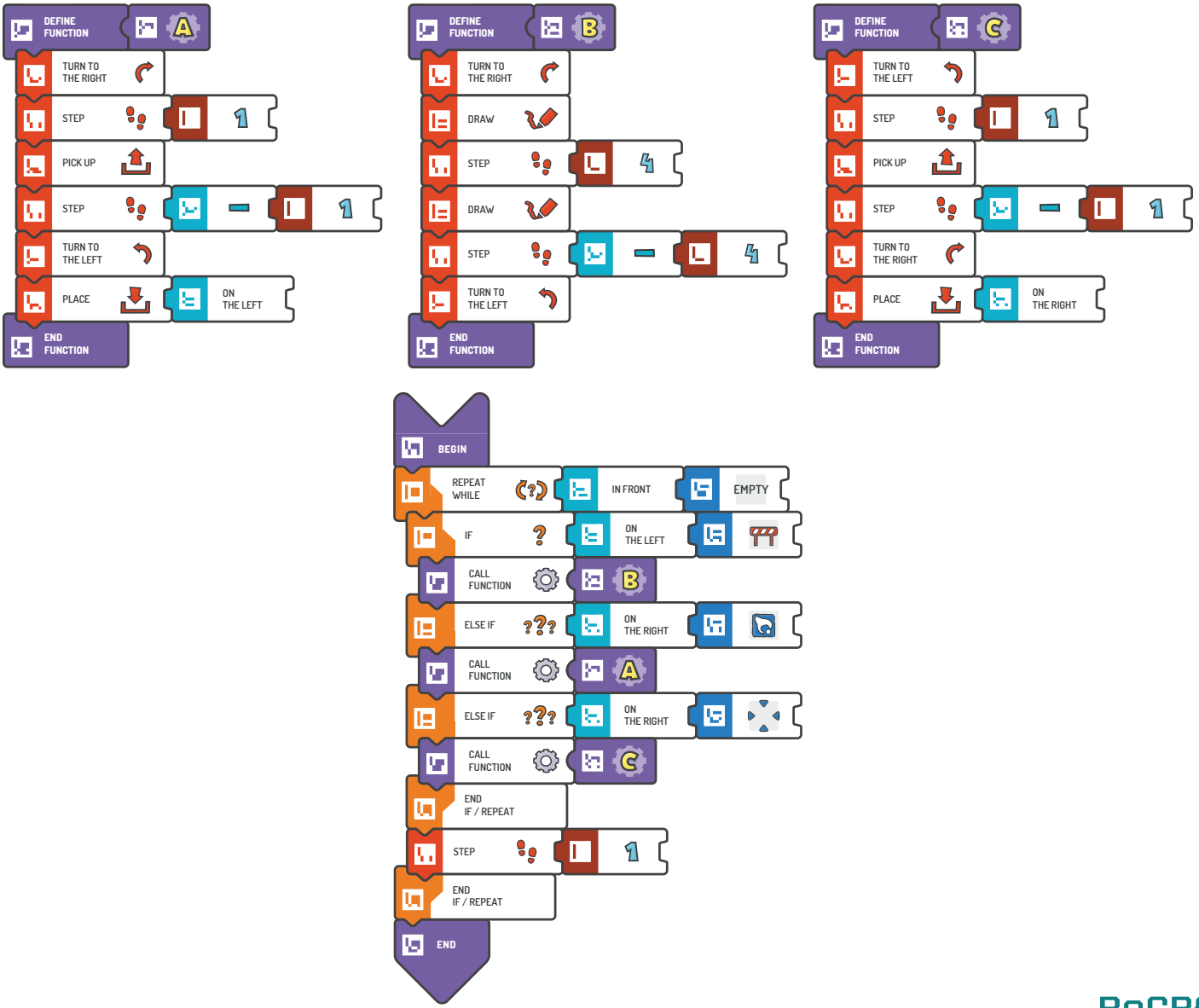


You can erase the function that has already been defined from your device's memory at any moment. Scanning a new definition for a function with a given name (e.g. A) is going to replace the definition that was stored in memory before.

Challenge:



Example solution:



Challenge:



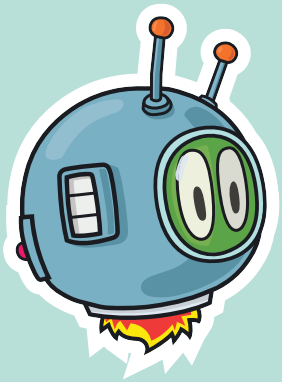
Example solution (in text format):

```
DEFINE FUNCTION A {
        TURN TO THE RIGHT
        STEP (1)
        PICK UP
        STEP (-1)
        TURN TO THE LEFT
        PLACE (ON THE LEFT)
}

DEFINE FUNCTION B {
        TURN TO THE RIGHT
        DRAW
        STEP (4)
        DRAW
        STEP (-4)
        TURN TO THE LEFT
}
```

```
DEFINE FUNCTION C {
        TURN TO THE LEFT
        STEP (1)
        PICK UP
        STEP (-1)
        TURN TO THE RIGHT
        PLACE (ON THE RIGHT)
}

BEGIN
        REPEAT WHILE (IN FRONT < EMPTY) {
                IF (ON THE LEFT < OBSTACLE ) {
                        CALL FUNCTION B
                } ELSE IF
                (ON THE RIGHT < COLLECTABLE OBJECT) {
                        CALL FUNCTION A
                } ELSE IF
                (ON THE RIGHT < ACTION SQUARE) {
                        CALL FUNCTION C
                }
                STEP (1)
        }
END
```



THANK YOU
ENJOY YOUR TIME WITH SCOTTIE GO!

Some of the presented solutions can be even shorter.

It's just the beginning of your fun with Scottie Go! Remember that you can use all expressions, commands and functions in any module.

If you have any remarks, comments or queries concerning this Teacher's Guide, email us at **contact@scottiego.com** and add the phrase "**TEACHER'S GUIDE**" in the title of your email.