Competencies and Concepts - Clarification

Coding

	Coding Competencies	Purpose
C1	Apply **computational thinking** skills to **develop** a set of logical instructions to solve a problem	Develop logical thinking & problem-solving skills (abstraction, decomposition, pattern recognition, algorithms (sequence of instructions)). Lead to C2
C2	Present a ***simple*** **coding solution** using symbolic or written statements representing sequences of commands, *single* repetition, and conditional constructs	Use and apply computational thinking and the problem-solving process to develop a solution for a coding problem. Involves C1, C3, C4, C5 (and by implication C6 and C7) Give **problem statement** *only* – learners must solve problem
C3	Interpret and execute a given symbolic or written set of commands	Read code, interpret code and execute the code (mimic software operations). Explain what it does (understand) Involves C1 Give code – learners must read, interpret, execute or read, interpret and explain
C4	Debug a given symbolic or written set of instructions.	Read code, understand what it does/explain what it does, find the error, correct the error, test again (repeat until it works correctly (bug-free)). Involves Involves C1, C3 Give code – learners must read, interpret and find the error, then correct the error OR When learners solve the problem (develop code – C2) they must test their code and make sure it works.
C5	Evaluate a given solution towards potential improvement.	Read code, determine what it does, improve/ optimise the code (e.g. use loops where required). Involves C1, C3 and often C4 and by implication C2 Give code – learners must make code shorter, find the shortest route (more efficient), change or extend (C2)
C6	**Recognise** and **interpret** patterns in symbolic sets of data or visualisations	Goal is generalising code to perform tasks depending on user input – NOT only for specific instances (and AI). Part of C1 and used withC5, C2 Also links to D8 and D9 Can be dealt with as part of C1
C7	**Create** or **complete** a pattern to represent a data set	

Robotics

	Robotics Competencies	Purpose
R1	Explain what a robot is in simple terms.	Theoretical aspects of Robotics. Answer questions such as What is a robot? What makes something a robot? What are the different parts of a robot? What is the function of each part? What are the different types of robots? Examples. Where are these used? What do robots do? How can they help us? Can they harm us? Can be done with R5 (progress as building and understand of robot's progress)
R2	Identify different types of robots	
R3	Outline the different components of a robot	
R4	Present an understanding of how robots affect the world	
R5	Design a simple artefact based on a set of design specifications	Use design thinking and the design process to develop robotic artefacts - Creative process
R6	Mimic the operations of a robot	Interpret & execute a set of robotic instructions (code) – **C3** in coding
R7	Create, test, and execute a set of robotic instructions	Write code to make a robot / artefact 'do' something Use computational thinking (C1) to develop code (R7 (C2) in coding)) for the robot Execute the code to test the code, correct the code, if necessary – **C2, C3, C4** in coding Note: **C1** also links to **C6** and **C7**

Digital Concepts

	Digital Concepts Competencies	Purpose
D1	Outline the concept of technology and purpose of information technology (IT).	What is technology? What is IT? Technology vs. IT? Etc.
D2	Recognise that he or she is living as a citizen in a digital world.	Digital citizenship. How must we behave as digital citizens in a digital world? What are the dangers, benefits, limitations?
D3	Demonstrate an understanding of the concept of a computing device.	What is a computing device? Examples Link to D1, D4, D5 and D7
D4	Identify the common uses of ICT in the real world.	What is ICT? T vs IT vs ICT? Examples. Link to D3, D5
D5	Differentiate between the components of an ICT system.	Components of an ICT system and their purpose/function Link to D3, D7
D6	Explain how the adaptation of technology impacted the world we work and live in.	How did technology change our world? E.g. at home, in school. Link to D2
D7	Present a basic understanding of the concept of input processing and output	Computing devices receives input, process the input and give a result or output. Link to computing devices (D3), D4 & D5, coding (C2) and robotics (R6, R7)
D8	Interpret a pattern to represent or communicate a message or image.	Encoding and Decoding – in FP, also links to C1, C6 and C7 E.g. decode an encoded pattern/message (D8) and encode a message/pattern (D9), e.g. decode an encoded message using a cipher
D9	Create a pattern to represent or communicate a message or image	

1.1 COMPETENCIES AND CONCEPTS: CLARIFICATION

1.1.1 Coding & Robotics CAPS Requirements per Grade per Term – Grade R – Grade 3

1.1.2 (The red is when a new strategy is added. When we add strategies, the cognitive load is bigger)

1.1.2.1 Grade R

Use big floor grids for learners to walk on. They can pack the arrows on the grid.

Term 1 • Do not move block-by-block – move till you reach an obstacle (Move like in a maze) • Use arrows – left, right, forward, back • No turns – turns are implied by the direction of the arrows • No repetitions (loops)	**Term 3** • Do not move block-by-block – move till you reach an obstacle (Move like in a maze) • Add moving block-by-block if learners are ready • Use arrows – left, right, forward, back • No turns – turns are implied by the direction of the arrows • No repetitions (loops)
Term 2 • Do not move block-by-block – move till you reach an obstacle (Move like in a maze) • Use arrows – left, right, forward, back • No turns – turns are implied by the direction of the arrows • No repetitions (loops)	**Term 4** • Do not move block-by-block – move till you reach an obstacle (Move like in a maze) • Add moving block-by-block if learners are ready • Use arrows – left, right, forward, back • No turns – turns are implied by the direction of the arrows • No repetitions (loops)

1.1.2.2 Grade 1

Use big floor grids for learners to walk on. They can pack the arrows on the grid.

Term 1 • Do not move block-by-block – move till you reach an obstacle (Move like in a maze) • Add moving block-by-block if learners are ready • Use arrows – left, right, forward, back • No turns – turns are implied by the direction of the arrows • No repetitions (loops)	**Term 3** • Move block-by-block • Use arrows – left, right, forward, back • Can use arrows or words, e.g. forward, back, left, right • No turns – turns are implied by the direction of the arrows • Use repetitions (loops) - only add repetitions after they packed out the whole code
Term 2 • Move block-by-block • Use arrows – left, right, forward, back • Can use arrows or words, e.g. forward, back, left, right • No turns – turns are implied by the direction of the arrows • Add repetitions (loops) - only add repetitions after they packed out the whole code	**Term 4** • Move block-by-block • Use arrows – left, right, forward, back • Can use arrows or words, e.g. forward, back, left, right • No turns – turns are implied by the direction of the arrows • BUT add a **turnaround**  for the sprite or character facing in the wrong direction at the start • Use repetitions (loops) - only add repetitions after they packed out the whole code

1.1.2.3 Grade 2

Use big floor grids for learners to walk on or use A2 or A3 grids on the tables. They must start packing the arrows below the grid by the end of the first term.

Term 1 • Move block-by-block • Use arrows – left, right, forward, back • Use arrows or words, e.g. forward, back, left, right • No turns – turns are implied by the direction of the arrows • Use a **turnaround**  if the sprite or character is facing in the wrong direction at the start • Use repetitions (loops) - only add repetitions after they packed out the whole code	**Term 3** • Move block-by-block • Use arrows or words – turn left, turn right, forward, turn around • Use left and right turn – they will now use turns and forward arrows • Use repetitions (loops)to shorten codes (NOT routes) • You can add other instructions as required by problems, e.g. jump, pick up, etc.
Term 2 • Move block-by-block • Use arrows and words – left, right, forward, turn around • Add turn left and turn right – they will now use turns and forward arrows • Use repetitions (loops)to shorten codes (NOT routes)	**Term 4** • Move block-by-block • Use arrows or words – turn left, turn right, forward, turn around • Use left and right turn – they will now use turns and forward arrows • Use repetitions (loops)to shorten codes (NOT routes) • You can add other instructions as required by problems, e.g. jump, pick up, etc. • Add a simple IF-statement (decision)

1.1.2.4 Grade 3

Use floor grids or A2, A3 or A4 grids on the tables. They must pack the arrows below the grid.

By the second term they can combine two constructs at a time, e.g. sequence and iteration (loop) or iteration (loop) and decision (IF-Statement)

Term 1 • Move block-by-block • Use arrows or words, e.g. forward, back, left, right, etc. • Use abbreviations if learners are at that level, e.g. Fwd., TL, TR, etc. • Use repetitions (loops) to shorten codes • Use turns • Use decision (IF-statement) • Use other instructions if required by the problem	**Term 3** • Same as term 1
Term 2 • Same as Term 1	**Term 4** • Same as term 1

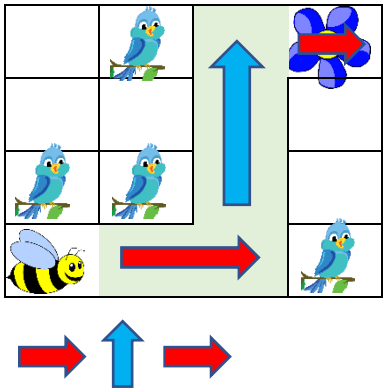
Give a problem statement: (Problem Statement 1)

The bee must go to the flower without alerting the bird. Do not walk in the same block as the bird. Step into the block with the flower.

Give a problem statement: (Problem Statement 2)

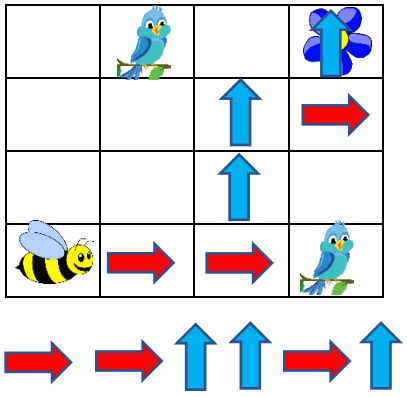
Find a safe route for the bee to go to the flower.

The first problem statement is easier, because you tell them what to avoid and how to get to the flower. The 2nd problem statement requires learners to apply logical and critical thinking



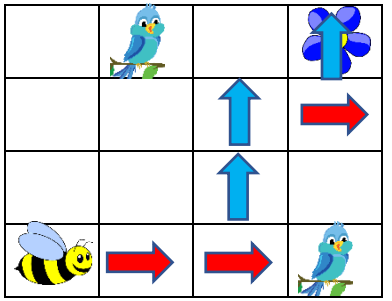
Grade R Term 1 and Term 2. Grade 1 Term 1 – IF LEARNERS WERE **NOT EXPOSED TO block-by-block movement IN GRADE R**

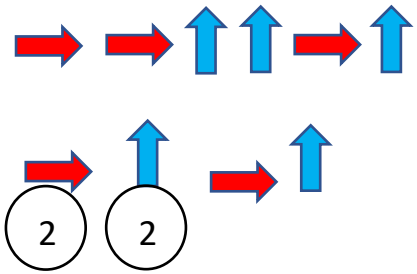
- Do not move block-by-block – move till you reach an obstacle (Move like in a maze).
- Use arrows – left, right, forward, back
- ONLY USE 1 ARROW IN ONE DIRECTION.
- They do not count blocks
- Arrows indicate DIRECTION
- If you pack it underneath the grid, it must still indicate the directions the bird must move.
- No turns – turns are implied by the direction of the arrows
- No repetitions (loops)



Grade 1 Term 1 – IF LEARNERS **WERE EXPOSED TO block-by-block movement IN GRADE R**

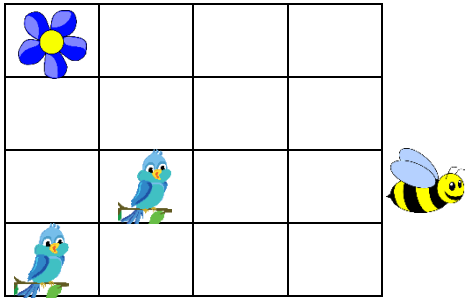
- Add moving block-by-block if learners are ready – If they did this in grade R and you are sure they are comfortable with it.
- Use arrows – left, right, forward, back
- Arrows indicate DIRECTION
- One arrow indicates one block moved
- If you pack it underneath the grid, it must still indicate the directions the bird must walk.
- No turns – turns are implied by the direction of the arrows
- No repetitions (loops)

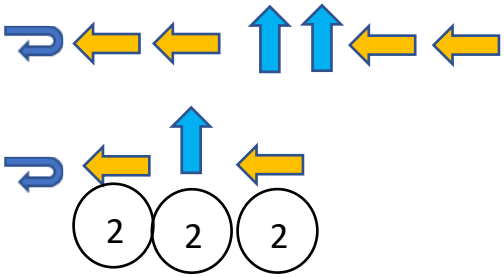




Grade 1 Term 2 – Learners move block-by-block.

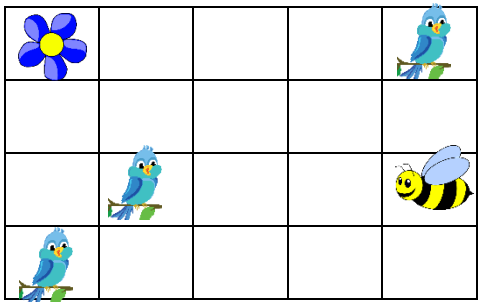
- Move block-by-block must now be implemented
- Use arrows – left, right, forward, back
- No turns – turns are implied by the direction of the arrows
- Add repetitions (loops)
- Only add repetitions after they packed out the whole code





Grade 1 Term 4 & Grade 2 Term 1 – Add the turn-around for when a character is facing the wrong direction and you need the character to turn-around to get onto the grid

- Move block-by-block
- Use arrows – left, right, forward, back
- No turns – turns are implied by the direction of the arrows
- Add repetitions (loops)
- Only **add repetitions** after they **packed out the whole code**



Grade 2 Term 2 to Grade 3 term 4 – Add the turn left / turn right and use **only the forward** arrows.

- Move block-by-block
- Use arrows – turn left, turn right and forward
- Add turns
- The coding solution MUST be packed BELOW the GRID, NOT on the grid

NB: **Arrows do not have to be in different colours.**

You can use arrows that are all the same colour. The direction the arrow indicates is important.