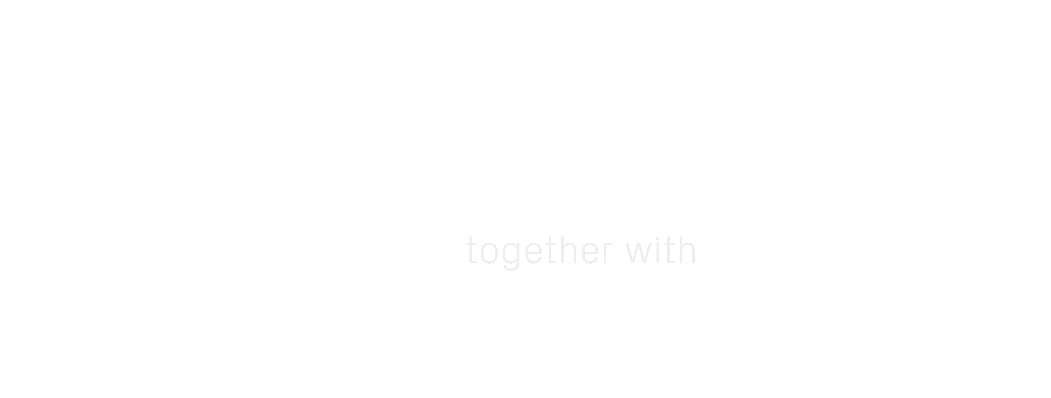
**digitalschoolhouse.org.uk dsh@ukie.org.uk**



Teaching Guide.

Fitness Frenzy

Introduction

This workshop was developed by Digital Schoolhouse together with Jo Hodge, Lower Key Stage 2 Phase Leader, Year 4 Teacher and Learning and Technologies Leader, Our Lady of Lourdes Catholic Nursery and Primary School, to incorporate computing into the curriculum in an innovative way.

In this workshop, students will use the Micro:bit to make their own pedometer with additional fitness workouts using different inputs.

Contents

[Learning Outcomes 3](#_Toc147156906)

[Files/Resources 4](#_Toc147156907)

[Session Overview 6](#_Toc147156908)

[Digital Schoolhouse Progression Matrix 15](#_Toc147156909)

[Algorithms 15](#_Toc147156910)

[Programming 16](#_Toc147156911)

[Data 17](#_Toc147156912)

[Hardware and Software 18](#_Toc147156913)

[Communication 19](#_Toc147156914)

[Digital skills 20](#_Toc147156915)

[Computing Programmes of Study Links 21](#_Toc147156916)

[Computational Thinking Strands 23](#_Toc147156917)

# Learning Outcomes

1. To design and build a pedometer using the accelerometer on a Micro:bit.
2. To add count-controlled loops and variables to code.
3. To write a simple algorithm for an exercise routine.
4. To use Micro:bit LEDs and inputs to design exercise routines.

# Files/Resources

|  |  |  |  |
| --- | --- | --- | --- |
| Filename | Resource Type | Purpose/Description | Activity No |
| Fitness Frenzy PPT | DSH-Fitness-Frenzy-Teaching-Presentation | Presentation for all of the activities and resource cards | ALL |
| Pedometer  Treasure Hunt Fact Cards | DSH-Fitness-Frenzy-Reference-Sheets | Double-sided fact cards with facts about pedometers for a treasure hunt unplugged activity. | 1 |
| Micro:bit Download Helpcard | Helpcard – DSH-Fitness-Frenzy-Teaching-Presentation slide 12 and 13 | Helpcard to support pupils when connecting Micro:bit to download programs. | 2 |
| Micro:bit LED Exercise Algorithm Worksheet | DSH-Fitness-Frenzy-Worksheet1-Exercise-Algorithm | Worksheet contained LEDS grids for pupils to design their exercise algorithm. |  |
| Pedometer Design and Evaluation Sheet | DSH-Fitness-Frenzy-Worksheet2-Strap-Design-and-Evaluation Sheet | Worksheet to enable pupils to design their strap for their oedometer and evaluate the finished product. | 4 |
| Pedometer Examples | DSH-Fitness-Frenzy-Example-Answers | Pedometer examples of finished product and related code. | 4 |
| Mini whiteboard and pen | Classroom resource | Used for students to complete the unplugged pedometer activity. | 3 |
| Weaving binary strap worksheet | DSH-Fitness-Frenzy-Worksheet3-Optional-binary-encoded-strap | Worksheet to enable students to weave a binary encoded strap for their pedometer.  One sheet between two. | 4 |
| Coloured paper | Classroom resource | Used for students to weave binary encoded straps for the optional unplugged activity. | 4 |
| Scissors | Classroom resource | Used for students to cut out strips of coloured paper for the optional unplugged activity. | 4 |

**PLEASE NOTE:** The activities outlined in this workshop pack are a suggested outline of how the workshop can be delivered. It is envisaged that teachers will adapt the resources and the organisation of them according to the needs of their class.

# Session Overview

This is a three-lesson unit of work linked to PE and DT to make a Micro:bit Pedometer. They will complete the following over the three activities:

* Learn all about pedometers and how they work presenting their findings. U/ B
* Explore the Micro:bit platform and code their pedometer using the shake function i.e. accelerometer B/I
* Learn about how a variable works using an unplugged activity U/B
* Design a strap and then test their completed pedometer I/A

**U = Unplugged activity, B = Beginner activity, I = Intermediate activity, A = Advanced activity**

**SESSION 1**

|  |  |  |
| --- | --- | --- |
| Activity No | Session Content / Activity | Resources Used |
| 1 What is a pedometer? | Outline activity: To understand what a pedometer is.  Present either plugged or unplugged.  Provide pupils with devices to research and in small groups they produce a poster.  Or have facts cards dotted around the room for them to go and collect ideas like a treasure hunt - unplugged activity. Each group has a picture to find with facts on the back.  In groups of 4, they are to present a poster or PowerPoint about what a pedometer is.  They need to find out:  What it does i.e. special features?  Are there different types?  What do they cost?  Any other fun facts about them…  Present either plugged or unplugged. Provide pupils with devices to research. Or can have facts cards dotted around the room for them to go and collect ideas. Give them 10-15 mins to produce their poster and then present findings. | DSH-Fitness-Frenzy-Teaching-Presentation pages 1 - 6  IPADS/ Computers for researching pedometers  Pedometer Treasure Hunt Fact Cards  Large flip paper and coloured pens |

**SESSION 2**

|  |  |  |
| --- | --- | --- |
| Activity No | Session Content / Activity | Resources Used |
| 2 Coding the Micro:bit pedometer | Outline activity: What is a Micro:bit –  Micro:bit - The micro:bit is a tiny computer. You can write programs for the micro:bit on your computer and then transfer them to the micro:bit to be run.  [**https://microbit.org/get-started/first-steps/introduction/**](https://microbit.org/get-started/first-steps/introduction/)  Code Tracing – Look at code on ppt. Give a copy to pupils in pairs and ask them to consider each question.  What will each block do?  What will happen on the start?  When you press button A, what will display on the Micro:bit?  Jot ideas on the sheet. Do they think it looks familiar? Establish a block-based code called Makecode - similar to Scratch - different colours for block functions. Discuss then run the code in Microbit.  Pupils to run and test the code to see if did what they predicted. Relate back that this is similar to Scratch. Work through slides or on Micro:bit website how to add blocks and then run their code. Show how the interface works if they have not used it previously using slide 11. Demonstrate how to connect the Micro:bit once so they can then just download the code more easily ( See helpcard).  Once they have run the code, discuss their original predictions - were they correct etc. | DSH-Fitness-Frenzy-Teaching-Presentation pages 7 – 14  Micro:bit download helpcard – DSH-Fitness-Frenzy-Teaching-Presentation slides 12-13  Micro:bits, USB cables and battery packs |

**SESSION 3**

|  |  |  |
| --- | --- | --- |
| Activity No | Session Content / Activity | Resources Used |
| 3 Unplugged pedometer activity  4 Designing the strap and testing design. | Outline activity: Designing, Coding, Testing and Evaluating their pedometer.  Recap what a pedometer is. Show information on slide:   * Step counter - helps us exercise and tell the time! * Can also be heart monitor and many more exciting things. * It is portable and counts steps by detecting the motion of our hands or hips - accelerometer.   Show slide 17 and explain to your students that they will need to use a variable to store the changing value of the numbers of steps that the pedometer has recorded while the program is running.  What is a variable?  What will change in our code that we will need to store?  Link back to scores in any games they have made in Scratch. Establish that the steps will change and be stored.  Ask your students what they will need a variable for in their pedometer program – they should say to store the number of steps.  Explain that variable names must be meaningful and ask your students to suggest what they will name this variable – they should suggest something along the lines of steps or numberOfSteps. You might like to talk about common ways of writing variable names here e.g. camelCasing and under\_scores and that students should not use spaces in their variable name.  **Unplugged activity.**  Put your students into pairs and hand out the mini whiteboard and pens. Instruct your students to write ‘steps’ on one side of the whiteboard – explain that the whiteboard is the variable, ‘steps’ is the variable name, and the blank side is where their data will be stored. Explain that one person in the pair should write down the number of steps their partner has taken as they walk around the room and update the data by rubbing out and replacing the number on their whiteboard. Run the activity for a few minutes, then the pair can swap over – remind your students to reset their variable by rubbing out the number on their whiteboard before starting the activity again.  Ask your students to explain how this activity demonstrates that a variable can change while a program is running – students should be able to say that the number changed while their partner was walking around.  Look at code on slide 18. Get them to add the code and explain that we need to add variable.  Ask the pupils to add the code to their micro:bit,  Once tested then ask what else could we add i.e. exercises.  Think back to first piece of code? Could we use some of this to add exercises? Could we use different inputs i.e. press button A, B or A and B etc. Different repeat loops e.g. count controlled loops? See child’s example code.  Complete the algorithm for each of their exercises buttons using DSH-Fitness-Frenzy-Worksheet1-Exercise-Algorithm.  Once their algorithm is complete, add their code to their pedometer code and test. Encourage pupils to code small sections first and then debug if needed.  Once coded, design a strap using DSH-Fitness-Frenzy-Worksheet2-Strap-Design-and-Evaluation Sheet.  To make their strap:   * Decorate a strip of white card or thick paper (or carry out the optional unplugged activity: Binary encoded straps). * Then reinforce with duct tape on the back. * Measure the card to wrist allowing 1 to 2 inches extra, * Cut off excess strap but do not throw away. * In the middle of the strap, tape the battery onto the reverse using the excess strap to hold (see pedometer examples) * Tape Micro:bit to the front. * Use a strip of duct tape to attach to the wrist.   **Optional unplugged activity:** Binary encoded straps  Begin by reminding your students that everything on a computer is processed in binary and then run the following weaving binary activity to create their pedometer strap.  Explain that students will be representing their name in binary using ASCII code for each letter (students should be aware of ASCII from previous work but if not a short explanation that every letter that you press on a keyboard must be converted into a number in binary and that ASCII is the set of numbers that each letter is represented by should be enough to begin the lesson).  1 Students cut along the dotted lines in their weaving binary sheet  2 With the coloured paper landscape in orientation, cut the coloured paper into strips  3 Students should look up the first letter of their name in their ASCII character set (see slide 24 of the teaching presentation) and plan their strap design using worksheet2.    Using one of the coloured slips, weave the binary for the first letter into the weaving binary sheet - passing the coloured paper under the strip in the worksheet for a 0 and over the strip for a 1 e.g.  Repeat the process for each remaining letter  Allow pupils time to then test out their pedometers by completing each of their exercises. Work with a partner and finally complete the evaluation section of the worksheet. | DSH-Fitness-Frenzy-Teaching-Presentation ppt pages 15 – END  Mini whiteboard and pen (one between two)  Micro:bits, USB cables and battery packs  DSH-Fitness-Frenzy-Worksheet1-Exercise-Algorithm  DSH-Fitness-Frenzy-Worksheet2-Strap-Design-and-Evaluation Sheet  DSH-Fitness-Frenzy-Worksheet3-Optional-binary-encoded-strap |

# Digital Schoolhouse Progression Matrix

The Digital Schoolhouse progression matrix is a simplified mechanism for measuring pupil progress. It stretches from base level understanding at the beginner level to introducing GCSE content at the advanced level. The shaded statements reflect skills and concepts covered in the workshop, these have been cross referenced to specific activities in order to reflect both their level of complexity and provide a guideline on which to measure progress. For more details about this framework see ‘Enter the Matrix’ included in this workshop pack.

### Algorithms

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Beginner** | **Activity No** | **Intermediate** | **Activity No** | **Advanced** | **Activity No** |
| Understanding | Understands what an algorithm is | 3/4 | Understands that algorithms are not the same as programming | 3/ 4 | Understands that different algorithms exist for the same problem | 3/ 4 |
| Writing | Represents algorithms using graphical notation such as pictures | 4 | Represents algorithms using structured notation such as flowcharts | 4 | Represents algorithms using pseudocode |  |
| Limitations | Understands that computers need precise instructions | 3/ 4 | Can identify tasks best completed by humans or computers | 3/ 4 | Understands that some problems cannot be solved computationally |  |
| Planning | Can identify the steps needed to solve a problem | 3/ 4 | Can identify the programming constructs needed to solve a problem (pattern recognition) | 3/ 4 | Can identify the modules needed to solve a problem e.g. top down design | 4 |
| Tracing | Uses logical reasoning to predict outputs and show an awareness of inputs |  | Uses logical reasoning to explain how an algorithm works | 3/ 4 | Evaluates the effectiveness of algorithms and models for similar problems | 3/ 4 |
| Designing | Designs solutions (algorithms) that use sequence, selection i.e. if, then and else and iteration |  | Designs solutions by decomposing a problem and creating a sub-solution for each of these parts | 3/ 4 | Designs a solution to a problem that uses generalization to create objects and classes (OOP) |  |

### Programming

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Beginner** | **Activity No** | **Intermediate** | **Activity No** | **Advanced** | **Activity No** |
| Writing | Can create a simple program in an environment that does not rely on text e.g. programmable robots etc | 2/ 4 | Has practical experience of a high-level textual language, including use of standard libraries |  | Has experience of designing programs that include a graphical user interface | 2/4 |
| Program flow | Understands that programs execute by following precise instructions | 2/3 /4 | Understands how modular programs work using sub-routines |  | Appreciates the effect of the scope of a variable e.g. a local variable can’t be accessed from outside its function unless passed as a parameter |  |
| Selection | Uses selection statements in programs, including an if, then and else statement |  | Understands the difference between, and appropriately uses if and if, then and else  Statements |  | Uses nested selection statements |  |
| Iteration | Uses loops, within programs | 2/3/4 | Understands the difference between, and uses ‘while’, ‘until’ and ‘for’ loops |  | Uses nested iteration and recursion |  |
| Debugging | Detects and corrects simple semantic errors i.e. debugging, in programs | 2/4 | Detects and corrects syntactical errors |  | Applies a modular approach to error detection and correction |  |
| Program design | Creates programs that implement algorithms to achieve given goals | 2/4 | Can design a program based on an algorithm | 2/3/4 | Designs modular programs using a range of methodologies e.g. RAD, waterfall |  |
| Data types and structures | Declares and assigns variables | 2/3/4 | Selects appropriate data types |  | Understands and uses one and two dimensional data structures |  |
| Operators | Uses arithmetic operators |  | Uses a range of operators and expressions e.g. Boolean |  | Understands and uses negation with operators e.g. not equal to |  |

### Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Beginner** | **Activity No** | **Intermediate** | **Activity No** | **Advanced** | **Activity No** |
| Representation | Recognises that digital content can be represented in many forms |  | Understands how bit patterns represent different forms of data e.g. character sets, sound, numbers and images |  | Understands how the same bit patterns can be used for different forms of data e.g. metadata |  |
| Transfer | Knows that data can be transferred from one computer to another |  | Knows that computers transfer data in binary |  | Understands and can explain the need for data compression, and performs simple compression methods |  |
| Types | Recognises different types of data: text, number |  | Defines data types: string, integer, real and Boolean |  | Understands how different data types can be used within data structures e.g. arrays must be made up of the same data type whereas lists can use several |  |
| Binary | Can carry out simple binary to decimal conversions | 4 | Performs operations using bit patterns e.g. binary addition, conversion between binary and hexadecimal, binary subtraction etc |  | Understands the relationship between binary and electrical circuits, including Boolean logic |  |
| File Size | Understands that data takes up space on a computer |  | Understands the relationship between binary and file size (uncompressed) |  | Knows the relationship between data representation and data quality e.g. resolution and colour depth etc, including the effect on file size |  |
| Data and Information | Understands the difference between data and information |  | Recognises that poor-quality data leads to unreliable results, and inaccurate conclusions |  | Understand the mechanisms used to cleanse data e.g. validation, range checks etc |  |
| Searching | Can sort data, use filters and perform single criteria searches for information |  | Queries data on one table using a typical query language, including more complex searches for information e.g. using Boolean and relational operators |  | Queries data on multiple tables using a typical query language |  |
| Structure | Recognises that data can be structured in tables to make it useful |  | Understands that all the data about a person or thing can be stored as a record |  | Knows what a relational database is, and understands the benefits of storing data in multiple tables |  |

### Hardware and Software

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Beginner** | **Activity No** | **Intermediate** | **Activity No** | **Advanced** | **Activity No** |
| Processing | Understands that computers have no intelligence and that computers can do nothing unless a program is executed |  | Knows that programs are executed by the processor i.e. the CPU |  | Understand that processors can work in different ways e.g. multitasking, scheduling |  |
| Software | Recognises that all software executed on digital devices is programmed |  | Knows that there is a range of operating systems and application software for the same hardware |  | Understands the concept of proprietary and open-source software including how this relates to licencing |  |
| Devices | Recognises that a range of digital devices can be considered a computer | 2/4 | Understands why and when computers are used | 2/3/4 | Understands how technology has developed e.g. Moore’s Law |  |
| Components | Recognises and can use a range of input and output devices | 2/4 | Recognises and understands the function of the main internal parts of basic computer architecture |  | Knows that processors have instruction sets and that these relate to low-level instructions carried out in the main internal parts of a computer |  |
| Operating systems | Understands that the operating system is software that specifies the function of a computing device |  | Understands the main functions of the operating system |  | Understands that there are different types of operating system and some of there common uses e.g. real time on auto pilot systems on a plane |  |
| Data transfer | Knows that data is transferred around a computer system using input devices, sensors and application software | 2/4 | Knows that data can be transferred between computer systems using physical, wireless and mobile  networks |  | Knows how data can be transferred between computer systems e.g. packet and circuit switching |  |
| Architecture | Understands the difference between hardware and software |  | Understands how hardware uses software to execute instructions e.g. the fetch-execute cycle |  | Understands computer architecture in relation to the fetch execute cycle, including how data is stored in memory |  |

### Communication

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Beginner** | **Activity No** | **Intermediate** | **Activity No** | **Advanced** | **Activity No** |
| WWW | Accesses content using a web browser |  | Understands that web pages are created using HTML and CSS |  | Understands how dynamic web pages use the client-server model and that web servers process and store data entered by users |  |
| Online safety | Understands why and how to keep personal information private and knows what to do when concerned about something online |  | Has an awareness of a range of online harms and demonstrates responsible use of technologies and online services in order to protect themselves |  | Understands how and why online threats are carried out and how to protect against them |  |
| Search engines | Navigates the web and can carry out simple web searches to collect digital content |  | Understands how to effectively use search engines e.g. Boolean, advanced search functions etc |  | Knows how search results are selected and ranked, including that search engines use ‘web crawler programs’ |  |
| Networks | Understands the difference between the internet and internet service e.g. world wide web |  | Understands data is transmitted between digital computers over networks, including different topologies e.g. ring, star, mesh |  | Knows the names and purposes of network components and protocols |  |
| Internet services | Shows an awareness of, and can use a range of internet services e.g. email |  | Selects, combines and uses internet services |  | Uses internet services to work collaboratively |  |

### Digital skills

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Beginner** | **Activity No** | **Intermediate** | **Activity No** | **Advanced** | **Activity No** |
| Invention | Uses software under the control of the teacher to create, store and edit digital content | 1/2/4 | Uses and selects internet services, digital devices and application software to create, store and edit digital content |  | Evaluates the appropriateness of digital devices, internet services and application software to achieve given goals |  |
| Audience | Understands what an audience is | 4 | Recognises the audience when designing and creating digital content | 4 | Undertakes creative projects that are tailored to meet the needs of an audience |  |
| Purpose | Can talk about how they use computers | 1/4 | Can talk about how other people use computers | 1/4 | Can discuss the issues around how other people might use computers e.g. Data Protection Act, Computer Misuse Act, Copyright etc |  |
| Evaluation | Can comment on the success of their solution | 4 | Designs and uses criteria to critically evaluate the quality of solutions |  | Documents user feedback, the improvements identified, and the refinements made to the solution |  |
| Content | Can gather content |  | Makes judgements about content when evaluating and repurposing it for a given audience |  | Evaluates the trustworthiness of content, considers the usability of visual design features and properties of media when designing and creating digital artefacts |  |

# Computing Programmes of Study Links

* 1. understand what algorithms are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions
  2. create and debug simple programs
  3. use logical reasoning to predict the behaviour of simple programs
  4. use technology purposefully to create, organise, store, manipulate and retrieve digital content
  5. recognise common uses of information technology beyond school
  6. use technology safely and respectfully, keeping personal information private; identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies
  7. design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
  8. use sequence, selection, and repetition in programs; work with variables and various forms of input and output
  9. use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs
  10. ~~understand computer networks including the internet; how they can provide multiple services, such as the world wide web; and the opportunities they offer for communication and collaboration~~
  11. ~~use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content~~
  12. select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information
  13. ~~use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact~~
  14. design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
  15. understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem
  16. use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions
  17. ~~understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers [for example, binary addition, and conversion between binary and decimal]~~
  18. ~~understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems~~
  19. ~~understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits~~
  20. undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users
  21. create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, design and usability
  22. understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns
  23. develop their capability, creativity and knowledge in computer science, digital media and information technology
  24. develop and apply their analytic, problem-solving, design, and computational thinking skills
  25. ~~understand how changes in technology affect safety, including new ways to protect their online privacy and identity, and how to identify and report a range of concerns.~~

# Computational Thinking Strands

AL – Algorithmic Thinking

|  |  |
| --- | --- |
| Ref. | Activity |
| A1 | Formulating instructions to achieve a desired effect |
| A2 | Formulating instructions to be followed in a given order (sequence) |
| A3 | Formulating instructions that use arithmetic and logical operations |
| ~~A4~~ | ~~Writing sequences of instructions that store, move and manipulate data (variables and assignment)~~ |
| A5 | Writing instructions that choose between different constituent instructions (selection) |
| A6 | Writing instructions that repeat groups of constituent instructions (loops/iteration) |
| ~~A7~~ | ~~Grouping and naming a collection of instructions that do a well-defined task to make a new instruction (subroutines, procedures, functions, methods)~~ |
| ~~A8~~ | ~~Writing instructions that involve subroutines that use copies of themselves (recursion).~~ |
| ~~A9~~ | ~~Writing sets of instructions that can be followed at the same time by different agents (computers/people, parallel thinking and processing, concurrency)~~ |
| ~~A10~~ | ~~Writing a set of declarative rules (coding in Prolog or a database query language)~~ |
| A11 | Using an appropriate notation to write code to represent any of the above |
| ~~A12~~ | ~~Creating algorithms to test a hypothesis~~ |
| ~~A13~~ | ~~Creating algorithms that give experience-based solutions (heuristics)~~ |
| ~~A14~~ | ~~Creating algorithmic descriptions of real world processes so as to better understand them (computational modelling)~~ |
| ~~A15~~ | ~~Designing algorithmic solutions that take into account the abilities, limitations and desires of the people who will use them~~ |

DE – Decomposition

|  |  |
| --- | --- |
| Ref. | Activity |
| D1 | Breaking down artefacts into constituent parts to make them easier to work with |
| D2 | Breaking down a problem into simpler versions of the same problem that can be solved in the same way (recursive and divide and conquer strategies) |

GE – Generalisation

|  |  |
| --- | --- |
| Ref. | Activity |
| G1 | Identifying patterns and commonalities in artefacts |
| G2 | Adapting solutions, or parts of solutions, so they apply to a whole class of similar problems |
| G3 | Transferring ideas and solutions from one problem area to another |

AB – Abstraction

|  |  |
| --- | --- |
| Ref. | Activity |
| Ab1 | Reducing complexity by removing unnecessary detail |
| Ab2 | Choosing a way to represent an artefact, to allow it to be manipulated in useful ways |
| ~~Ab3~~ | ~~Hiding the full complexity of an artefact (hiding functional complexity)~~ |
| ~~Ab4~~ | ~~Hiding complexity in data, for example by using data structures~~ |
| ~~Ab5~~ | ~~Identifying relationships between abstractions~~ |
| ~~Ab6~~ | ~~Filtering information when developing solutions~~ |

EV – Evaluation

|  |  |
| --- | --- |
| Ref. | Activity |
| E1 | Assessing that an artefact is fit for purpose |
| E2 | Assessing whether an artefact does the right thing (functional correctness) |
| ~~E3~~ | ~~Designing and running test plans and interpreting the results (testing)~~ |
| E4 | Assessing whether the performance of an artefact is good enough (utility: effectiveness and efficiency) |
| ~~E5~~ | ~~Comparing the performance of artefacts that do the same thing~~ |
| ~~E6~~ | ~~Making trade-offs between conflicting demands~~ |
| E7 | Assessing whether an artefact is easy for people to use (usability) |
| E8 | Assessing whether an artefact gives an appropriately positive experience when used (user experience) |
| ~~E9~~ | ~~Assessment of any of the above against the specification and set criteria~~ |
| ~~E10~~ | ~~Stepping through processes or algorithms/code step-by-step to work out what they do (dry run/tracing).~~ |
| ~~E11~~ | ~~Using rigorous argument to justify that an algorithm works (proof)~~ |
| ~~E12~~ | ~~Using rigorous argument to check the usability or performance of an artefact (analytical evaluation)~~ |
| ~~E13~~ | ~~Using methods involving observing an artefact in use to assess its usability (empirical evaluation)~~ |
| ~~E14~~ | ~~Assessing whether a product meets general performance criteria (heuristics)~~ |