**Scenario 1 (Storytelling)**

One pedagogy for teaching computational thinking concepts is storytelling. The aim of this part of the workshop is to demonstrate how storytelling can be used as a pedagogy to teach CT concepts. This resource is based on materials developed by Computing At Schools (CAS) UK and available from the Teaching London Computing website[[1]](#footnote-1).

The task is to design a presentation on selected CT concepts based on the activity.

**Introduction**

Locked-in syndrome is a medical condition that can leave a person’s intelligent mind locked inside a useless body, able to sense everything but unable to communicate.

As there is no cure for locked-in syndrome there isn’t a lot that medics can do beyond making their patients comfortable. Or is there? (How could computational thinking skills be used?)

‘The Diving Bell and the Butterfly’ is the autobiography of Jean-Dominique Bauby, written after he woke up in a hospital bed totally paralysed. In the book, he describes life with locked-in syndrome.

Put yourself in his position after waking up in the hospital bed. How could you communicate? How could you write a whole book? You have only a helper with a pen and paper to write down your ‘words’? All you can do is blink one eye. You can’t move in any other way. That means you can’t speak. Use the space provided to suggest a way to communicate?

|  |
| --- |
|  |

[An initial idea might be to use morse code (short and long blinks) or map 1 blink for an A, 2 blinks for a B etc. The helper counts the blinks and writes down the letter. This is a core aspect of computational thinking called *algorithmic thinking*:]

Algorithmic thinking is the type of thinking that leads to algorithms. An *algorithm*[[2]](#footnote-2) is a list of rules to follow in order to solve a problem.

A *protocol* is a special type of algorithm used to pass information between two people or computers. Because the algorithm in this example is used to pass information between Bauby and his helper, it is an example of a protocol.

This algorithm has two parts – one for Bauby to follow and one for the helper.

Algorithms are a way of capturing intelligence and sharing it with others. Once developed, the necessary intelligence to solve a problem is encoded in the algorithm.

The user(s) of the algorithm (in this case Bauby and the helper) need not have any understanding of what they are doing – they just need to be able to follow the instructions (e.g. count the blinks and write down the letter)

**Key Point**

Algorithmic thinking results in a general solution (in the form of steps) as opposed to a single answer. The power of algorithms is that they provide us with general solutions to problems.

**Improving our initial algorithm**

So far we have a way to map blinks to letters – what about spaces, what about punctuation marks, capital letters? Another important aspect of Computational Thinking (and algorithmic thinking) is *evaluation* i.e. testing (checking the details – what happens if the person blinks by mistake?) and thinking of ways to improve existing solutions.

|  |
| --- |
| Use the space provided to suggest improvements to turn blinks into letters involving the helper being the first part of the protocol. |

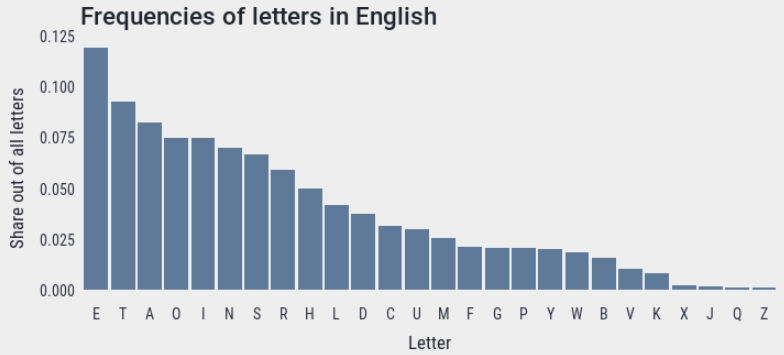
Examples might be

* Bauby blinks when the letter he is thinking of is read aloud by the helper (who starts at A each time). Try this with your neighbour. Communicate your initials. What are the best/worst case scenarios?
* Helper can guess the word (how can we use *pattern matching* to generalise this approach to invent predictive texting? Often problems turn out to be essentially the same as something you’ve already seen/solved in a different situation – pattern matching is the skill of spotting that a new situation is essentially the same as something you’ve seen before – essentially we are searching for a letter – this is the general form of the problem we are trying to solve).

**Frequency Analysis**

Frequency analysis has been used to crack secret codes throughout history. One example comes from the movie, ‘The imitation game’ in which a machine was programmed to decode words based on other words already known to exist in certain messages.

|  |
| --- |
| How might the frequency analysis of letters shown below be used to improve the algorithm in this story? |



Aside: What CT skills do you think would be required in order to produce the above chart?

**Using abstraction as a tool to evaluate our solution**

Is the ‘new improved’ algorithm better? (What do we mean by better?) How fast is it?

|  |
| --- |
| Use the space below to compare the ‘speed’ of the original algorithm with the one based on frequency analysis.  Hints:   1. Use abstraction to ignore any unnecessary details 2. Consider the best and worst case scenarios 3. Now consider the average case |

|  |
| --- |
| Describe how you used the concept of abstraction to help? |

It is possible to reduce the worst case to five questions per letter - read on to see how!

**Decomposition (Divide and Conquer)**

Let’s start with a guessing game.

Guess the number I am thinking of – between 1 and 100 - in 20 questions or less

A good strategy is to ask questions that rule out half the numbers each time.

e.g. Is is greater than 50? Yes

e.g. Is is greater than 75? No

e.g. Is is greater than 62? No

e.g. Is is greater than 56? Yes

etc

*Analysis*

Let’s say the number I was thinking of was between 1 and 1,000,000. What would be the minimum number of questions to narrow it down to one number.

Half the numbers can be ruled out with each question. After one question, we are down to 500,000 numbers left, two questions 250,000, then 125,000, then about 64,000 possibilities (simplifying a little to make the numbers easier!), 32,000, 16,000, 8000, 4000, 2000, 1000… After 10 questions there are only 1000 numbers left out of the original million it could be. Keep going…500 left after another question, 250, 125, 64 (ish) 32, 16, 8, 4, 2 and on the 20th question there is only one possibility left. So with the right questions, in the worst case it takes only **twenty** questions to find the number.

|  |
| --- |
| Explain how the divide-and-conquer technique just described can be transferred to the context of this story i.e. to find one of the 26 letters of the alphabet. |

|  |
| --- |
| Draw a decision tree showing questions to ask to get any letter of the alphabet in five questions or less. |

|  |
| --- |
| Use the decision tree to design an encoding system for unique codes (made up of 1s and 0s) for each letter |

**Understanding People (as part of computational thinking)**

Computational thinking is about solving problems for people. People therefore come first. You have to understand the problem you are solving from their point of view, before you dream up solutions. Otherwise your great technical solution could be useless. To be a great computer scientist, you have to understand people.

Perhaps we should have started with the person. Were we counting the right thing?

As our measure of work – our ‘abstraction’ – we used the number of questions asked. That is the job of the helper and it may be tedious but it’s not difficult. What if blinking was a great effort for Bauby. His solution involved him blinking only once per letter. Our divide-and-conquer algorithm requires him to blink five times. Multiply that by a whole book. We could have made it five times harder.

It could be blinking is easy and our algorithm is better. We don’t know the answer, because we didn’t ask the question. We should have asked first. We should have started with the person.

Furthermore, Bauby’s original solution is easy for anyone to walk in and understand. Ours is more complex to follow and might need some explaining before the visitor understands and Bauby is not going to be the one to do the explaining. Thinking about people is important!

|  |
| --- |
| Final question  Could technology be used to implement the algorithm? If so how and what should be taken into account? |

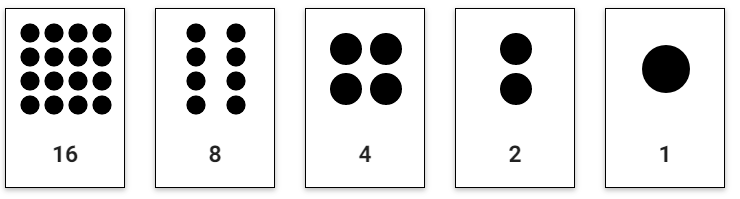
**Reflection**

Reflect of how the computational thinking concepts listed below were demonstrated in the story. Incorporate these concepts into your design.

|  |  |
| --- | --- |
| **Term** | **Example** |
| Algorithm | |
| Abstraction | |
| Decomposition | |
| Pattern Recognition and Generalisation | |
| Evaluation | |

**Scenario 2 (Working in Binary – A kinaesthetic activity)**

By using five individual cards based the illustration below devise a kinaesthetic classroom activity that could be used to demonstrate the binary number system



|  |
| --- |
| List some questions that could be asked during the activity.  e.g. What cards would be need to turn face down so that the (decimal) number 13 is displayed? |

|  |
| --- |
| What other concepts could this activity be used to teach?  e.g. Other number systems? Modulo arithmetic (mod ) |

|  |
| --- |
| Discuss how a machine based on an imaginary binary currency (i.e. valid coins are 1, 2, 4, 8, and 16 units) might work. Give examples. |

|  |
| --- |
| Describe how the activity could be used to explore the computational thinking concept of **abstraction** at an introductory level, e.g. how do we represent binary digits? |

|  |
| --- |
| Describe how the activity could be used to explore the computational thinking concept of **decomposition** at an introductory level, e.g. what are the individual pieces that make up the whole number? |

|  |
| --- |
| Describe how the activity could be used to explore the computational thinking concept of **pattern recognition** at an introductory level, e.g. what happens to a binary number when we append a 0? A 1? |

|  |
| --- |
| Describe how the activity could be used to explore the computational thinking concept of **logic** at an introductory level, e.g. what happens when we double a binary number |

|  |
| --- |
| Describe how the activity could be used to explore the computational thinking concept of **algorithmic thinking** e.g. write a sequence of steps to convert from a binary number to a decimal number. What if the binary number was being entered on the fly? |

|  |
| --- |
| Describe how the activity could be used to explore the computational thinking concept of **evaluation,** e.g. how many different values can be represented by a given number of bits? |

|  |
| --- |
| Describe how the activity could be used to explore the computational thinking concept of **data representation,** e.g. how can letters (and other symbols) be represented in binary? |

|  |
| --- |
| What extensions to this activity do you think would be appropriate for LCCS? |

**Further Reading**

1. The Evolution of Computers in Society (NCCA)

<https://curriculumonline.ie/getmedia/a5e0d88d-e0f1-43bc-ab68-349b5660fbce/NCCA-The-Evolution-of-Computers-in-Society-LC-SC.pdf>

TinyURL: <https://tinyurl.com/y7jkx4cd>

1. Computer Science Field Guide ([University of Canterbury](http://www.canterbury.ac.nz/), New Zealand)

<http://www.csfieldguide.org.nz/en/chapters/data-representation.html>

1. Locking a Dead Man’s Chest

<http://www.cs4fn.org/binary/lock/>

**Scenario 3 (Magic using Card Tricks)**

One pedagogy for teaching computational thinking concepts is magic. The aim of this part of the workshop is to demonstrate how card tricks can be used as a pedagogy to teach CT concepts. This resource is based on materials developed by Computing At Schools (CAS) UK and available from the Teaching London Computing website

Task:

Read the tricks described below carefully and practice performing them using the cards provided. Both tricks are examples of self-working tricks (aka algorithms).

Design a presentation that demonstrates the use of magic as a pedagogy for CT concepts

**Invisible Palming[[3]](#footnote-3)**

Start with fifteen cards. Get a volunteer to spread their ten fingers on a table as though playing the piano. Ask the audience to chant ‘two cards make a pair’ and as they do so, place the cards in pairs between the volunteers’ fingers. Place the final card between the last two fingers saying ‘one card left over’ as you do so.

Ask the audience to chant ‘two cards make a pair’ again and as they are chanting, take back the pairs placing them into two separate piles. Take the last card saying ‘we have one card left over’. Place this one extra card on one of the piles saying ‘that pile now has the extra card’.

Explain that you are going to use invisible palming to move the extra card from the pile you placed it on, to the other pile. Place your hand (face down) over the extra card and rub it as though you are making the card disappear – then lift your palm and show the audience that the card has indeed become invisible. Now move your hand to the other pile as though you are depositing the card there.

Finally reveal that the extra card has indeed moved piles by counting out the pairs in each pile!

|  |
| --- |
| Use the trick to explain algorithms and algorithmic thinking. Decompose this trick as much as possible. |

|  |
| --- |
| Try to think of as many examples as possible where computer algorithms are used to help people in everyday life. |

**Australian Magician’s Dream[[4]](#footnote-4)**

Instructions

1. Setup. Place a known card (e.g. 8 of Hearts) in the 16th position from the top of a deck and another known card (e.g. Ace of Spaces) at the 32nd position. Put another 8 of Hearts (from a second pack) and leave it aside in an envelope.
2. Discard roughly the bottom half of the pack. Spread the cards face up on the table and ask a volunteer to point roughly where they think the middle card is (for the trick to work they must point at a card between the 16th and 32nd positions). Discard all cards below the selected card and lift the remaining cards into your hands (the 8 of Hearts is still the 16th card from the top)
3. Repeatedly discard every second card. Now deal out the cards in sequence making two piles – add every other card to the alternate pile. Start the first pile with the first card and start the second pile with the second card. All odd numbered cards will go on the first pile and all even numbered cards will go on the second pile. While doing this say “Down” as you place the cards face down in the first pile and “Under” as you place the cards face up in the second pile. Once all the cards have been dealt discard the “Down” pile. Repeat the process with the cards from the “Under” piles until only one card remains – the 8 of Hearts!
4. Reveal the contents of the envelope.

|  |
| --- |
| Decompose the above instructions above into a more precise algorithm |

|  |
| --- |
| Use logical thinking and abstraction (and binary numbers) to explain how this trick works |

**Reflection**

Reflect of how the computational thinking concepts listed below were demonstrated in the tricks. Incorporate these concepts into your design.

|  |  |
| --- | --- |
| **Term** | **Example** |
| Algorithm | |
| Abstraction | |
| Decomposition | |
| Pattern Recognition and Generalisation | |
| Evaluation | |

**ADDITIONAL NOTES:**

**Scenario 4 (Knights Tour Puzzle)**

Another pedagogy for teaching computational thinking concepts is through puzzles. The aim of this part of the workshop is to demonstrate how a puzzle known as Knight’s Tour can be used as a pedagogy to teach CT concepts. This resource is based on materials developed by Computing At Schools (CAS) UK and available from the Teaching London Computing website[[5]](#footnote-5).

To get started you will need to browse to <https://tinyurl.com/yawsgv9d>[[6]](#footnote-6) and download and unpack the file ‘TouristAgency.ZIP’

Task: The task is to adapt the resources into a suitable lesson on computational thinking concepts. Focus is on pattern matching, abstraction, data representation, decomposition and algorithmic thinking.

|  |  |  |
| --- | --- | --- |
| **Problem 1**  A knight in chess can move either two squares horizontally and one square vertically or two squares vertically and one square horizontally. You must find a sequence of moves that starts from square 1, visits every square exactly once by making a knight’s moves, and finishes where it started.  Try a brute-force approach first. | |  |
| **Problem 2**  You are a hotel tour guide and must figure out a route that starts (and ends) at the hotel that visits every tourist site exactly once |  | |

**NOTES**

|  |
| --- |
| Which of the two problems did you find easier and why? |

|  |  |
| --- | --- |
| Study the two diagrams below and explain how they relate:   1. to each other and 2. to the problems outlined earlier | |
|  |  |

**Reflection**

Reflect on how the computational thinking concepts listed below were demonstrated in this scenario. Incorporate these concepts into your design.

|  |  |
| --- | --- |
| **Term** | **Example** |
| Pattern Recognition and Generalisation | |
| Data Representation | |
| Abstraction | |

**ADDITIONAL NOTES:**

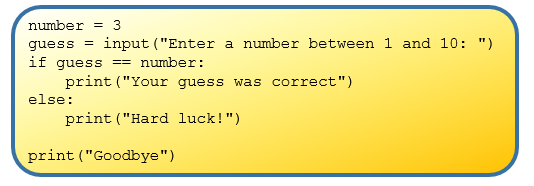
**Scenario 5 (Role play)**

Browse to <https://teachinglondoncomputing.org/resources/inspiring-unplugged-classroom-activities/the-brain-in-a-bag-activity/> and watch the video entitled ‘brain in a bag’

|  |
| --- |
| Describe how the computational thinking concepts of abstraction, decomposition and algorithmic thinging are addressed in the video. |

|  |
| --- |
| Identify the any programming constructs addressed in the video |

Using ideas you have gleaned from the video create the program below[[7]](#footnote-7) using people to represent instructions and the resources provided (string, baton etc).



|  |
| --- |
| Suggest ways in which this activity could be extended to re-inforce computational thinking concepts and programming constructs |

1. https://teachinglondoncomputing.org/free-workshops/computational-thinking-searching-to-speak/ [↑](#footnote-ref-1)
2. For an excellent introduction to algorithms watch the BBC4 documentary entitled *The Secret Rules of Modern Living: Algorithms* at https://www.youtube.com/watch?v=kiFfp-HAu64 [↑](#footnote-ref-2)
3. Reference: https://teachinglondoncomputing.org/resources/inspiring-unplugged-classroom-activities/the-invisible-palming-activity/ [↑](#footnote-ref-3)
4. Reference: https://teachinglondoncomputing.org/resources/inspiring-unplugged-classroom-activities/the-australian-magicians-dream-activity/ [↑](#footnote-ref-4)
5. https://teachinglondoncomputing.org/resources/inspiring-computing-stories/computational-thinking-knightstour/ [↑](#footnote-ref-5)
6. Alias for https://nccacurriculum.azurewebsites.net/Senior-cycle/Senior-Cycle-Subjects/Computer-Science/Support-Material-draft-for-T-L/1-Computational-Thinking/Unplugged-CT [↑](#footnote-ref-6)
7. From Python Skills Workshop, PDST, May 2018 [↑](#footnote-ref-7)