



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Leaving Certificate Computer Science National Workshop 7

Day 1





Workshop Overview

Session 1 10:00 - 11:30	Computational thinking IV
Tea/Coffee 11:30 – 12:00	
Session 2 12:00 - 13:30	Formative Assessment for LCCS
Lunch 13:30 - 14:30	
Session 3 14:30 - 16:30	Computer systems IV



Key Messages

Leaving Certificate Computer Science aims to develop and foster the learner's creativity and problem-solving, along with their ability to work both independently and collaboratively

LCCS can be effectively mediated through the use of a constructivist pedagogical orientation which will incorporate participatory and inquiry-based learning activities (whole-class, group, pair or individual).

Digital technologies used in LCCS have the potential to enhance collaboration, learning and reflection, by enabling students to learn more efficiently and to facilitate work that might not otherwise be possible.



Supports Provided by Oide

National
Workshops

Webinars

School Support

Scoilnet

Skills
Workshops

Collaboratives

Oide website

CompSci



CPD Supports



Mentoring



Computers in Education Society of Ireland
Cumann Ríomh-Oideachais na hÉireann



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

NW7 Session 1: Computational Thinking V



LEAVING CERTIFICATE
COMPUTER SCIENCE





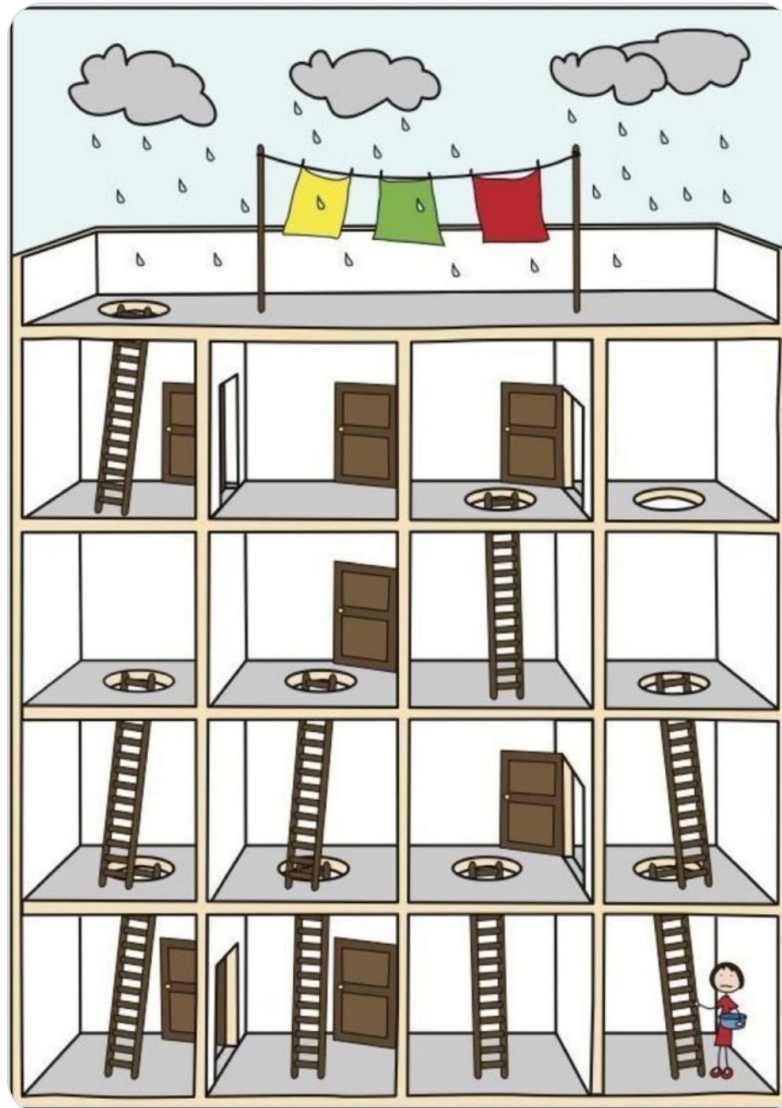
Overview of the session

Part 1	Computational Thinking Concepts
Part 2	Computational Thinking Activities



WARM UP

How many ladders to reach the top? 

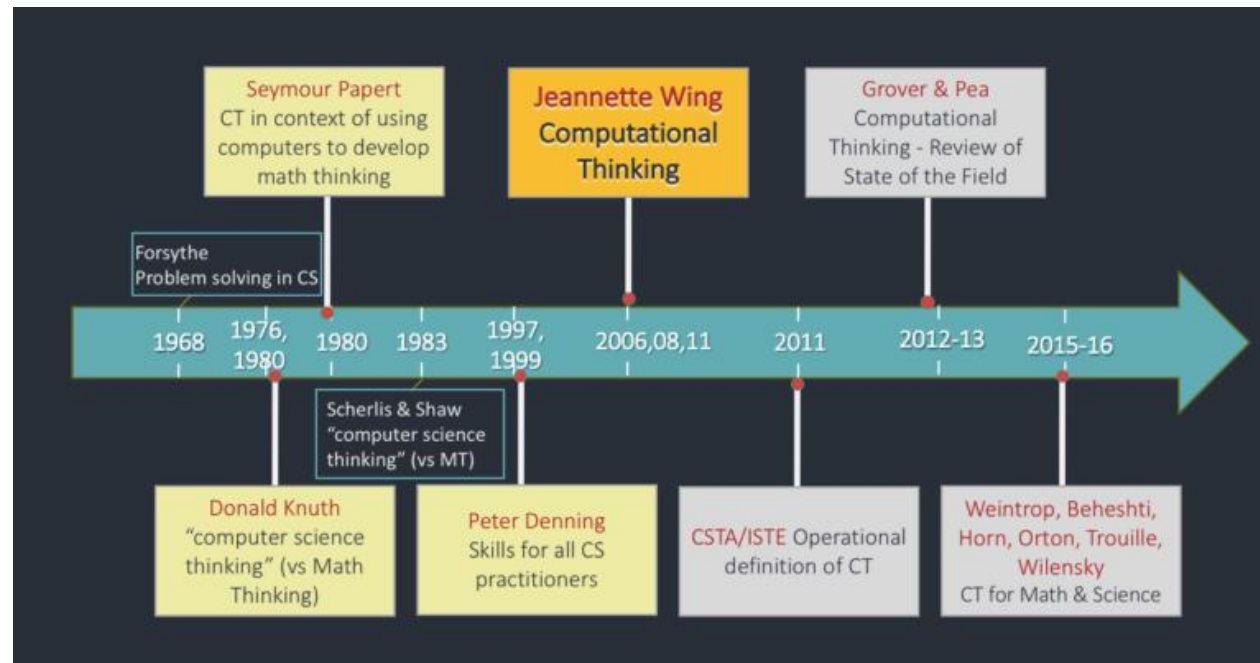




What is Computational Thinking?

"Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent."

Jeannette M. Wing, Carnegie Mellon University (2011)

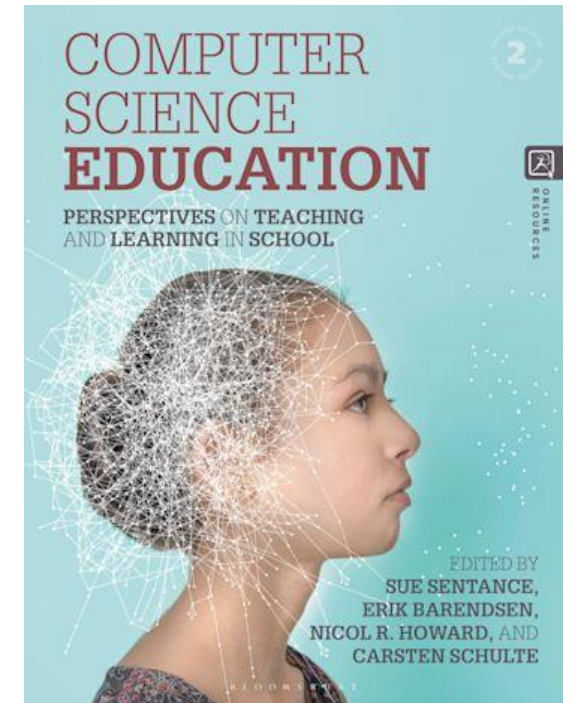


<https://www.shuchigrover.com/a-tale-of-two-cts-and-a-revised-timeline-for-computational-thinking/>

Computational Thinking Concepts



Source: <https://csunplugged.org/en/computational-thinking/>





Computational Thinking: Daily Examples

- Looking up a name in an alphabetically sorted list (Linear search: start at the top; Binary search: start in the middle)
- Queueing at a bank, supermarket, check-in desk, passport control (Performance analysis of task scheduling)
- Taking your children to football, music and the swimming pool (Traveling salesman with more constraints)
- Cooking a gourmet meal (Multi-tasking, Parallel processing)
- Cleaning out your garage (Keeping only what you need vs. throwing out stuff when you run out of space)
- Storing away your child's toys scattered on the floor (using hashing e.g., by shape, by color)



Why is Computational Thinking important?

- It moves students beyond being technologically literate
- It creates problem solvers instead of software technicians
- It emphasises the creation of knowledge rather than the use of information
- It presents endless possibilities for creative problem solving
- It enhances the problem-solving techniques you already teach

(Source: Pat Phillips, NECC 2007, Atlanta)

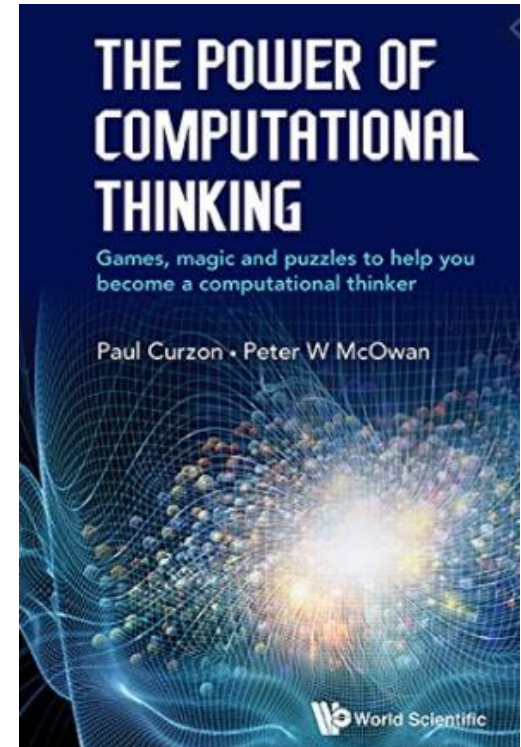


“What are effective ways for teaching computational thinking?”



How to teach Computational Thinking?

- ✓ Increase your own CT knowledge
- ✓ Integrate CT concepts into everyday instruction
- ✓ Use CT terms for everyday tasks e.g. “*Let’s create an algorithm for...*”
- ✓ Encourage students to formulate and test their own hypotheses e.g. “*Crime rates are on the rise...*”
- ✓ Provide opportunities for students to transfer their learning to other situations





Computational Thinking and Problem Solving

- The **four pillars model** (algorithms, pattern recognition, decomposition, abstraction) is well suited to the constructivist ethos of Leaving Certificate Computer Science
- **György Pólya's four principles** (understand, plan, carry out, look back) of **problem solving** is popular among maths educators
- **Wales/Woods model** (define the situation, identify the goal, generate ideas, plan, act, review)
- **Bransford IDEAL process**: Identify, Define, Explore, Anticipate and Act, Look.



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Applying Computational Thinking Skills



LEAVING CERTIFICATE
COMPUTER SCIENCE



Looking at Marriage

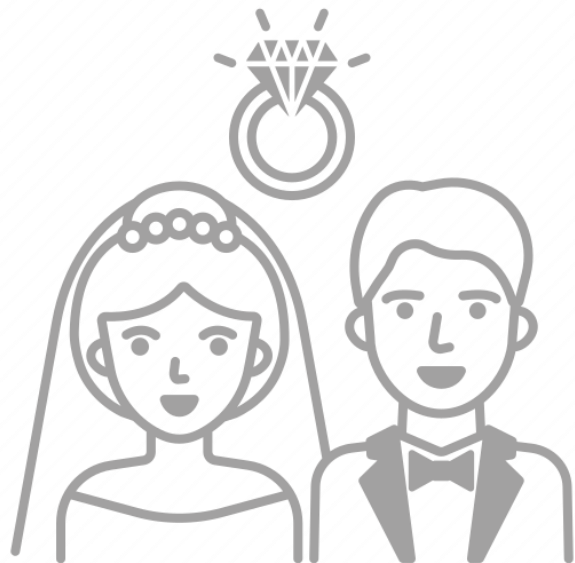


- Jack is looking at Anne. Anne is looking at George.
- Jack is married. George is not married.
- Is a married person looking at an unmarried person?

	Married	Looking At
Jack	Yes	Anne
Anne	?	George
George	No	?

Conclusion: Yes, a married person is looking at an unmarried person.

Q. Does it matter whether Anne is married or not?



	Married	Looking At
Jack	Yes	Anne
Anne	YES	George
George	No	?

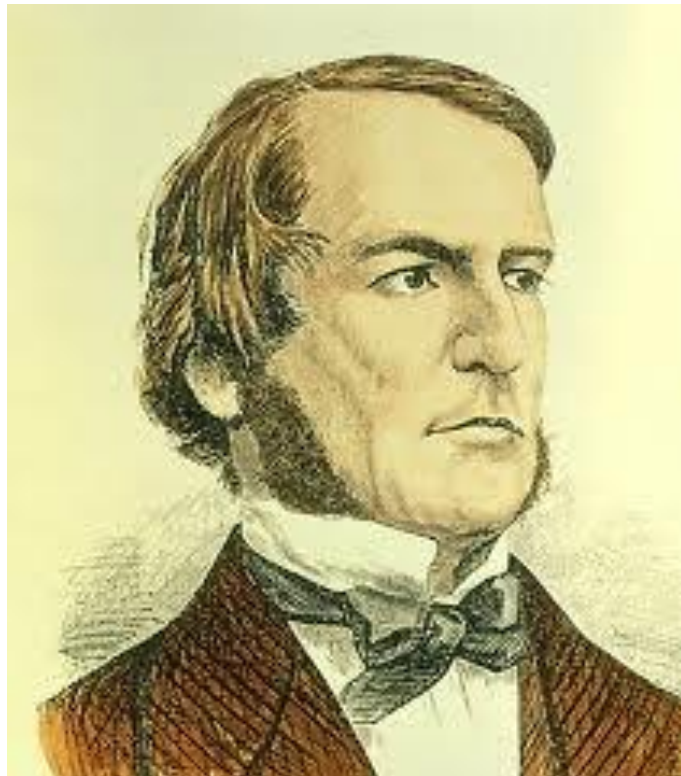
Anne is married and looking at George who is unmarried

	Married	Looking At
Jack	Yes	Anne
Anne	NO	George
George	No	?

Jack is married and looking at Anne who is unmarried



George Boole



- Born 1815 in the English cathedral city of Lincoln. Died in 1864.
- Inventor of Boolean Logic, which is the basis of modern digital computers.
"An Investigation of the Laws of Thought"
- First Professor of Mathematics at Queen's College Cork (now UCC).

⊕ Logic



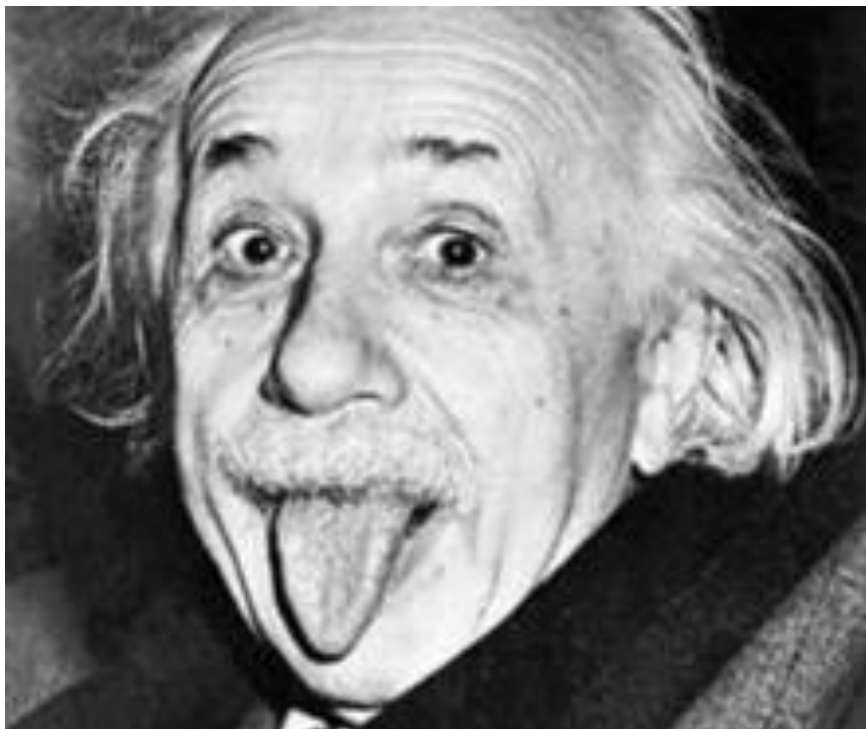


Group Activity





Einstein's Riddle



Source: <http://mathscircles.ie/#!/boole2school-lessons>

There are four bungalows in our cul-de-sac, numbered 1 through 4 from left to right. They are made from these materials: straw, wood, brick and glass.

- Mrs. Scott's bungalow is somewhere to the left of the wooden one and the third one along is brick.
- Mrs. Umbrella owns a straw bungalow.
- Mr. Tinsley does not live at either end but lives somewhere to the right of the glass bungalow.
- Mr. Wilshaw lives in the fourth bungalow, and the first bungalow is not made from straw.

Each person lives in a different house made of a different material from all the others.

Who lives where, and what is each person's bungalow made from?



Halloween Puzzle

During Halloween, four strange characters visited a certain school: a witch, a goblin, a ghost, and a black cat.

Each of them went into exactly one of these rooms: classroom 2, classroom 3, classroom 4 and the staff room.

- The goblin stole a notebook.
- The cat painted her paws.
- The ghost hid in a desk.
- The witch left a present.



Use the clues provided to determine which classroom each character visited.

Clue #1: Nothing was stolen from classroom 4.

Clue #2: The ghost hid either in classroom 2, or in the staff room.

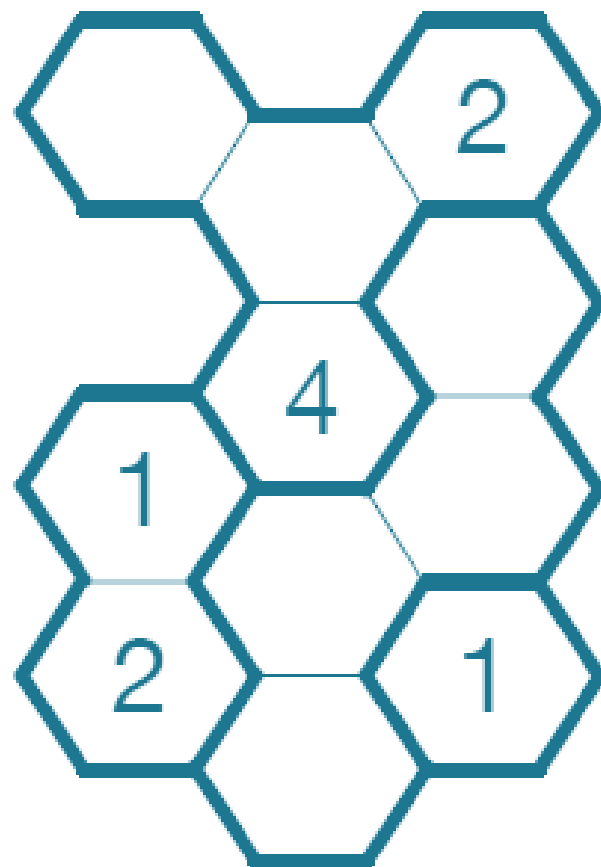
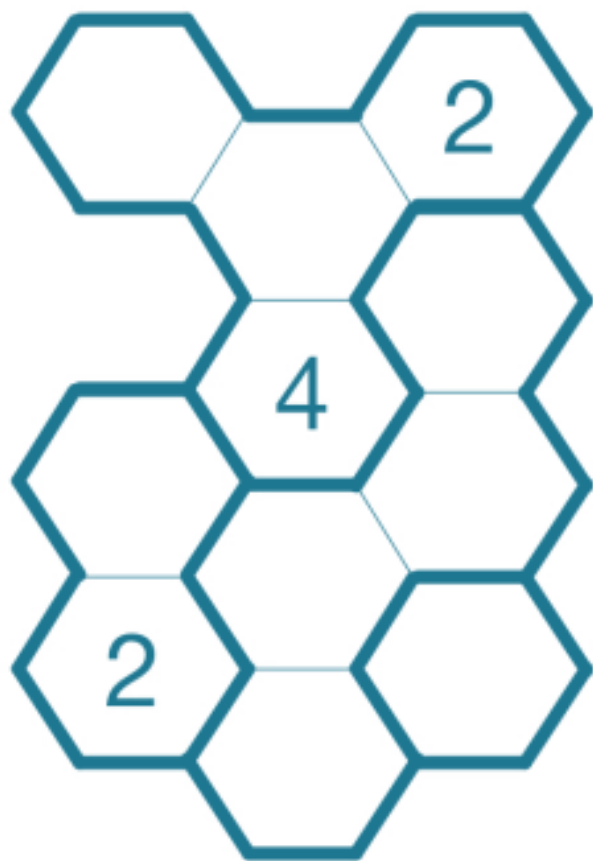
Clue #3: Classroom 2 was not visited by a goblin.

Clue #4: No notebooks or paints are ever kept in the staff room.

Clue #5: The black cat did not prowl through classroom 4.

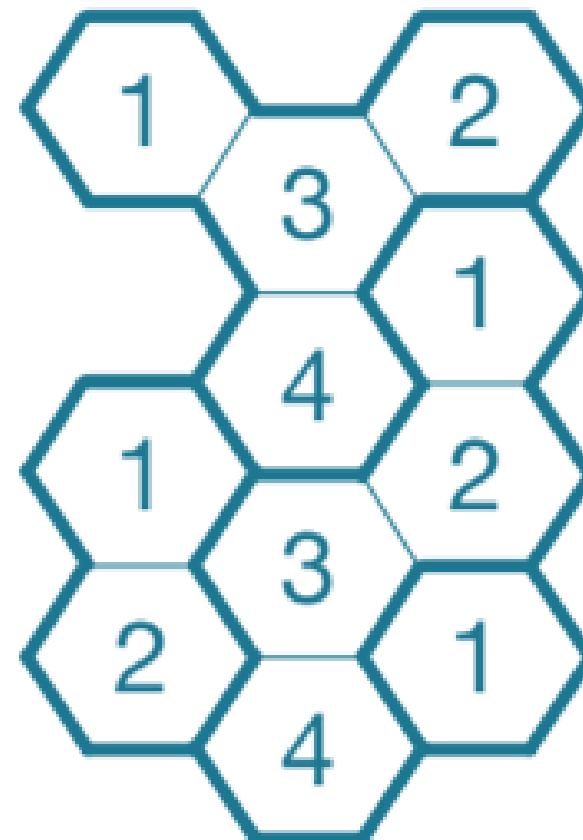
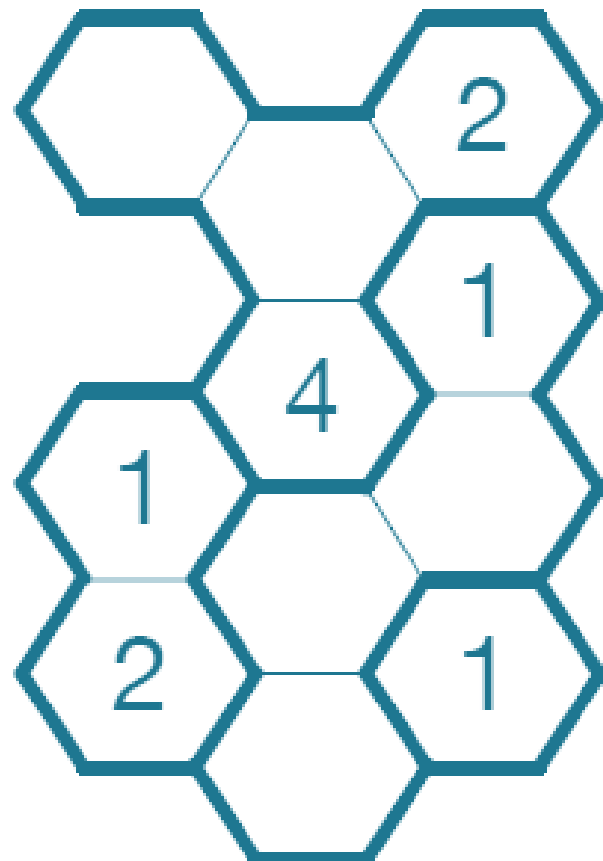
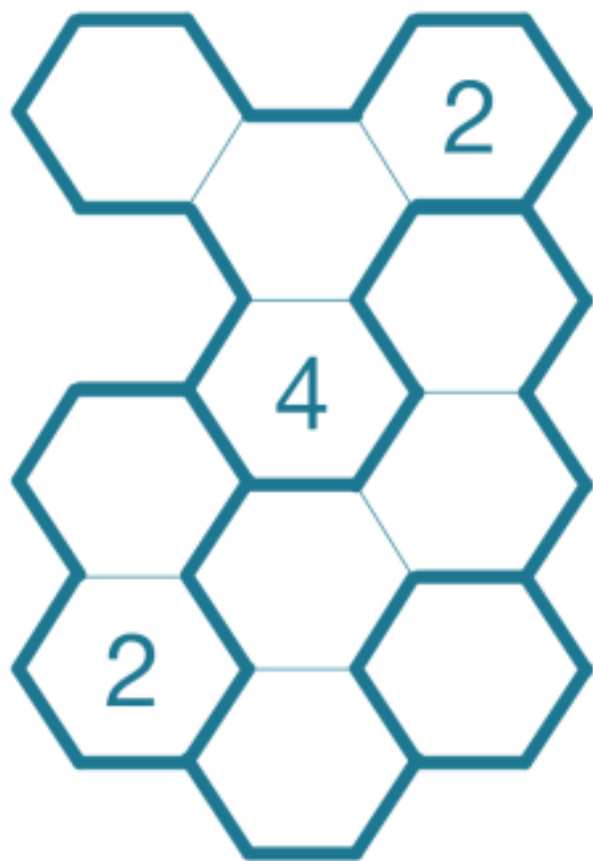


Cut Hive Logic Puzzles



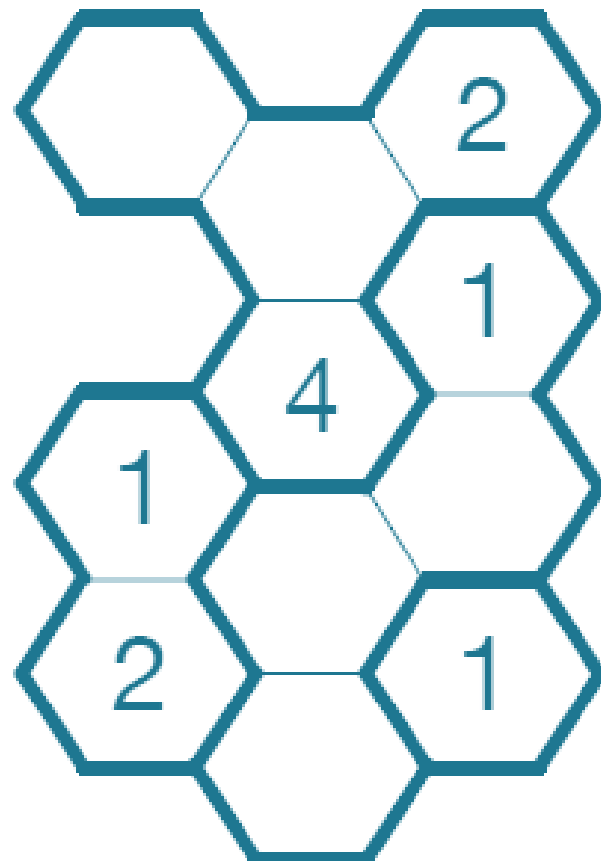
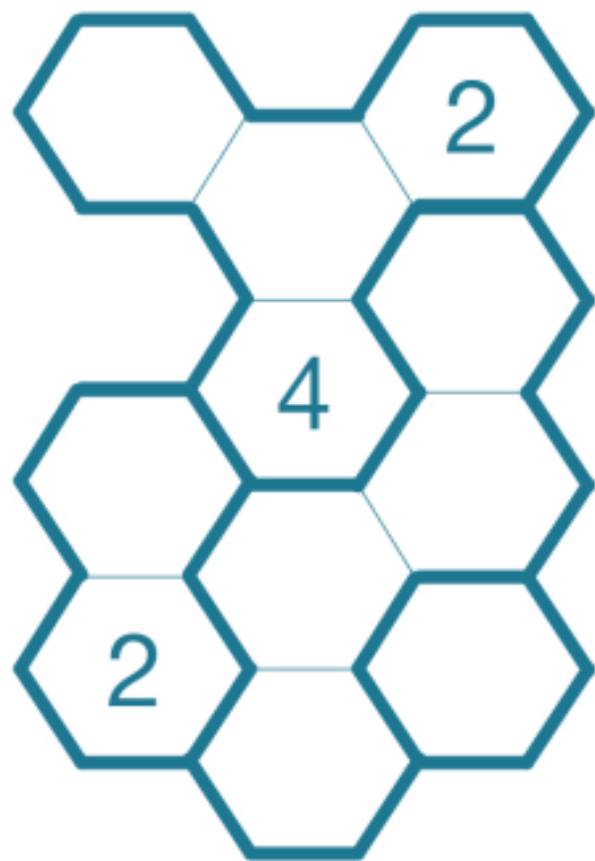


Cut Hive Logic Puzzles



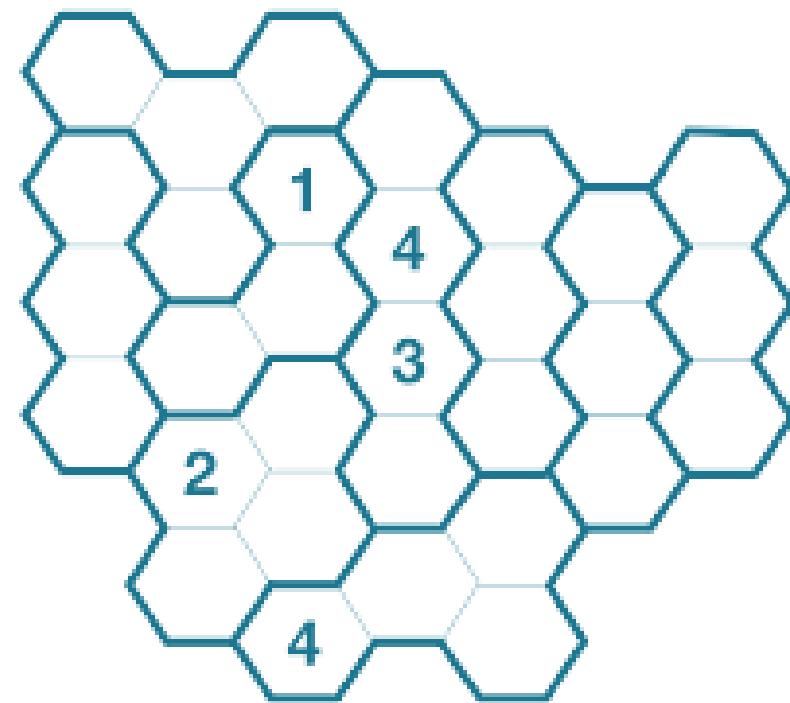
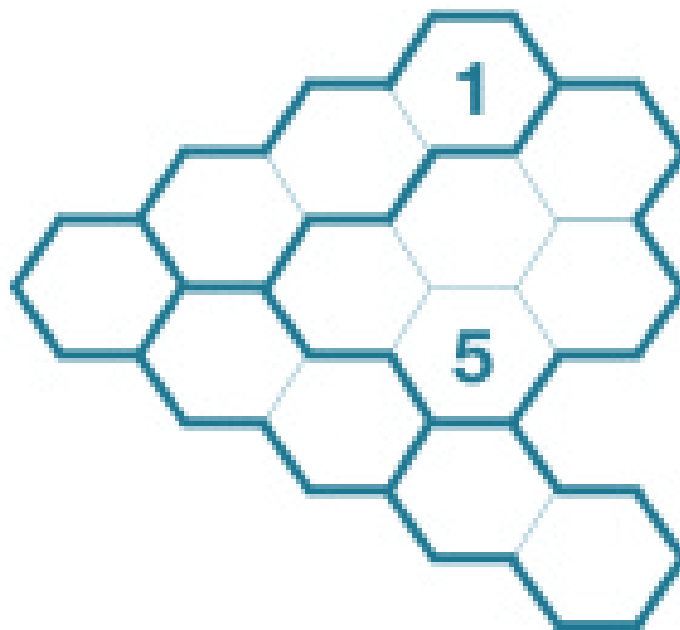
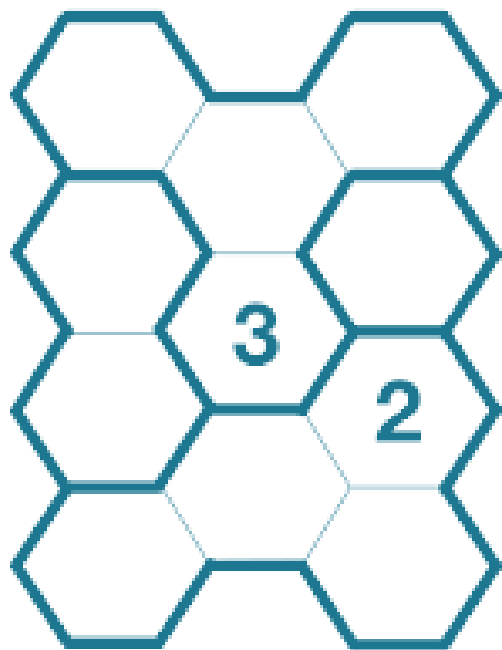


Cut Hive Logic Puzzles



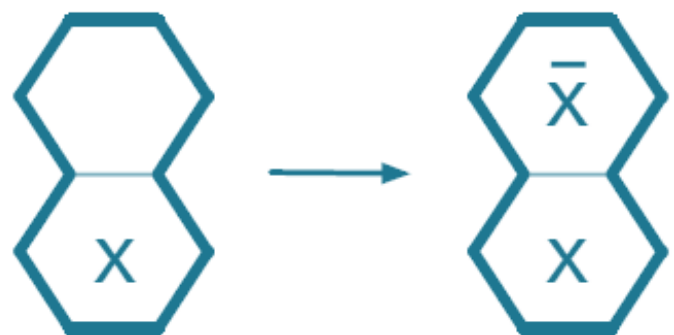


Cut Hive Logic Puzzles

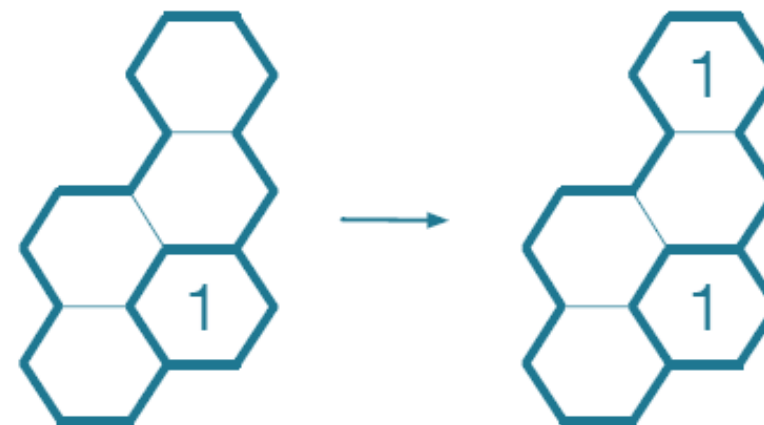




Cut Hive Logic Puzzles



Single Hexagon



Corners

⊕ Generalising and patterns

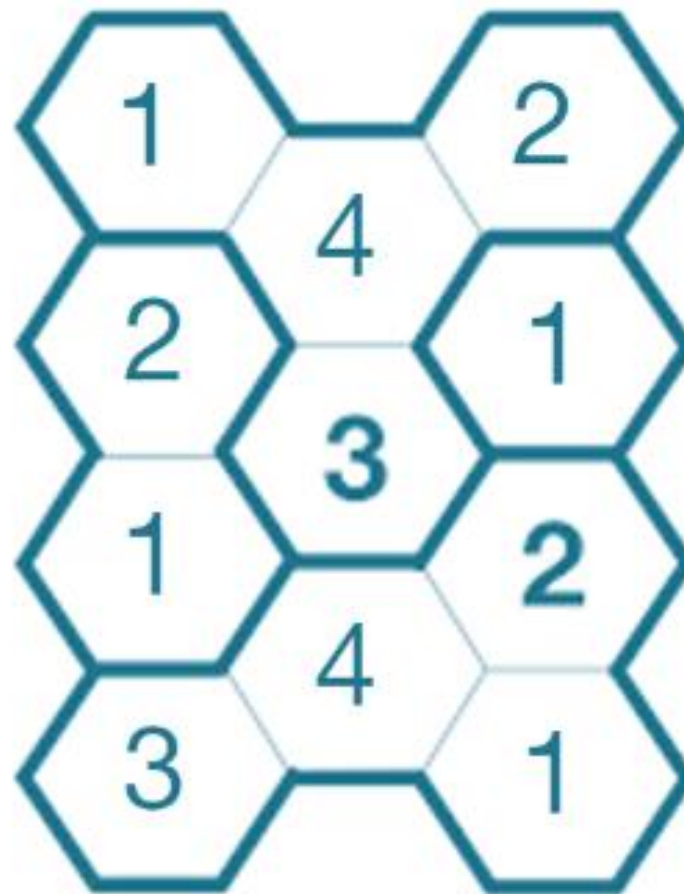


⊕ Abstraction



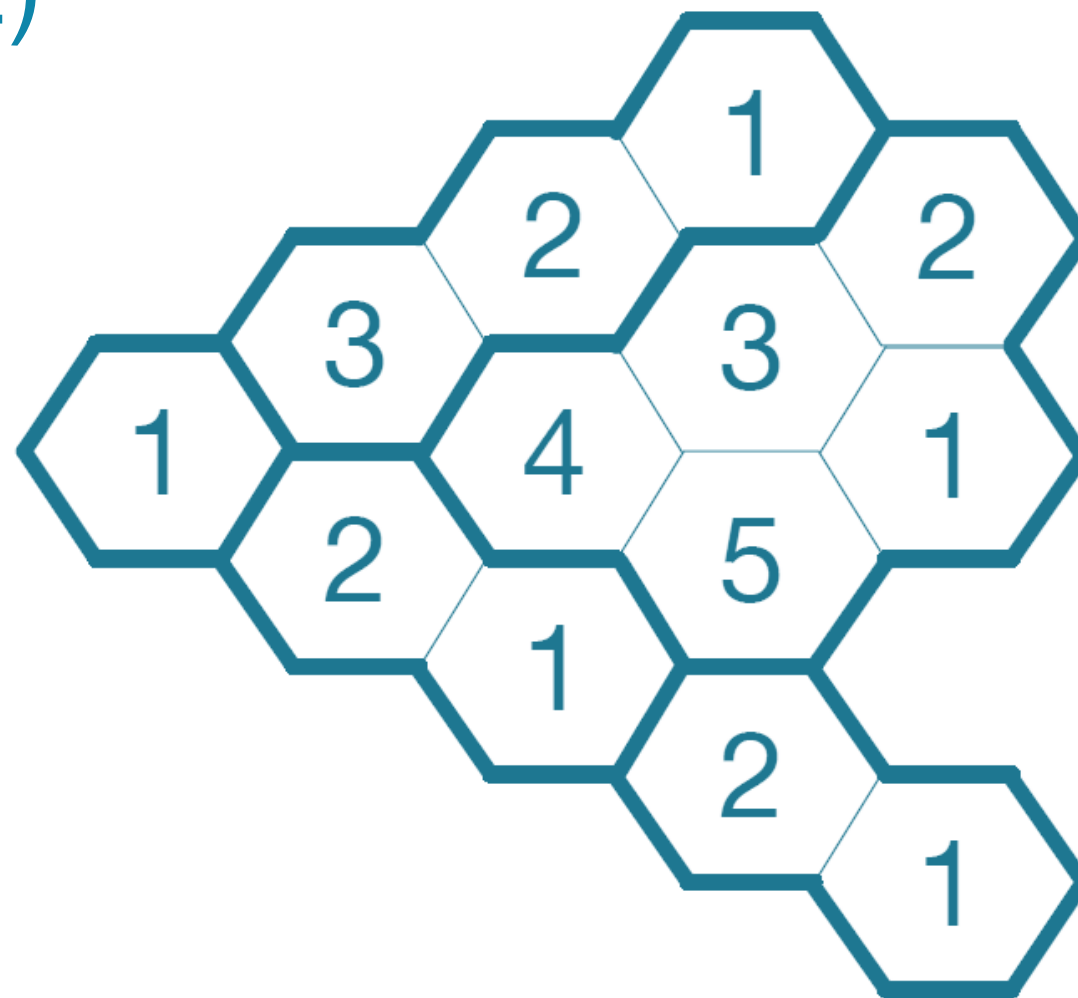


Solution (no1)



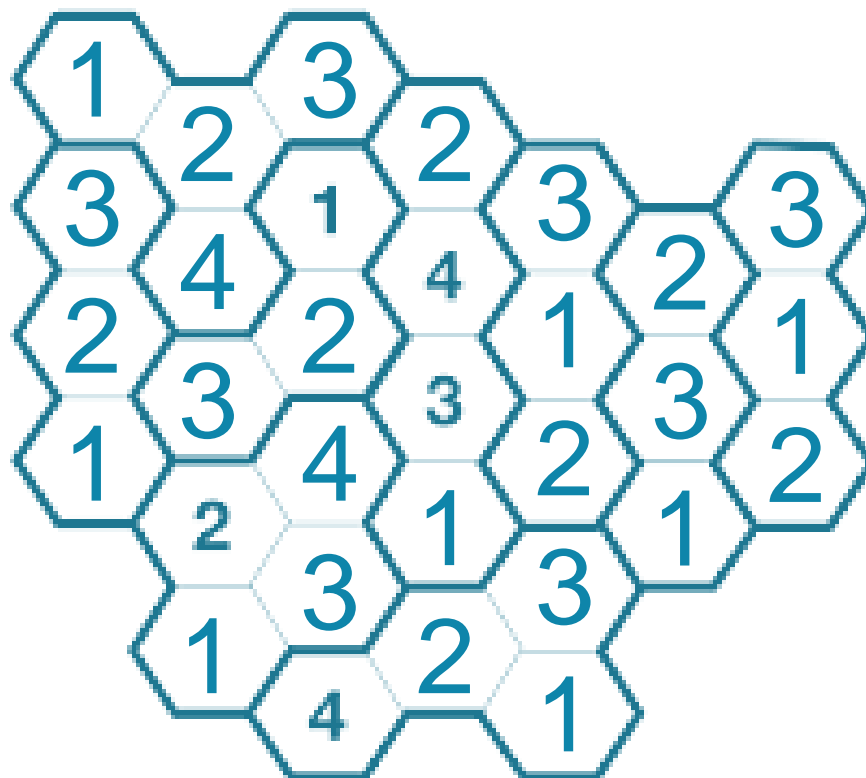


Solution (no2)





Solution (no3)





Bertrand's Box Problem

We have three boxes:



Box #1



Box #2



Box #3

- One box contains two gold coins
- One box contains two silver coins
- One box contains one gold coin and one silver coin



Bertrand's Box Problem



Shuffle the boxes so you don't know which is which

Pick a box and take out a coin.

If it is a gold coin, what is the probability that the other coin in that box is also gold?



Bertrand's Box Problem

- Work by yourself for about 5 minutes.
- Work towards a solution, try to create a representation for your solution, get an answer and a reason for your answer.
- After 5 minutes talk about your answer, your solution and your reasoning with the others on your table.
- See if you can come to a consensus on the answer.
- After each group has had time to discuss, we will discuss together and see if we can reach agreement on the answer.





Bertrand's Box Problem



<https://youtu.be/CGMc8B60ZpU>



Algorithmic Thinking

The aim is to swap the positions of the black and white pieces.



Pieces can move either by sliding into an adjacent empty square, or by jumping a single adjacent piece into the empty square immediately beyond.



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Group Activity

Role play





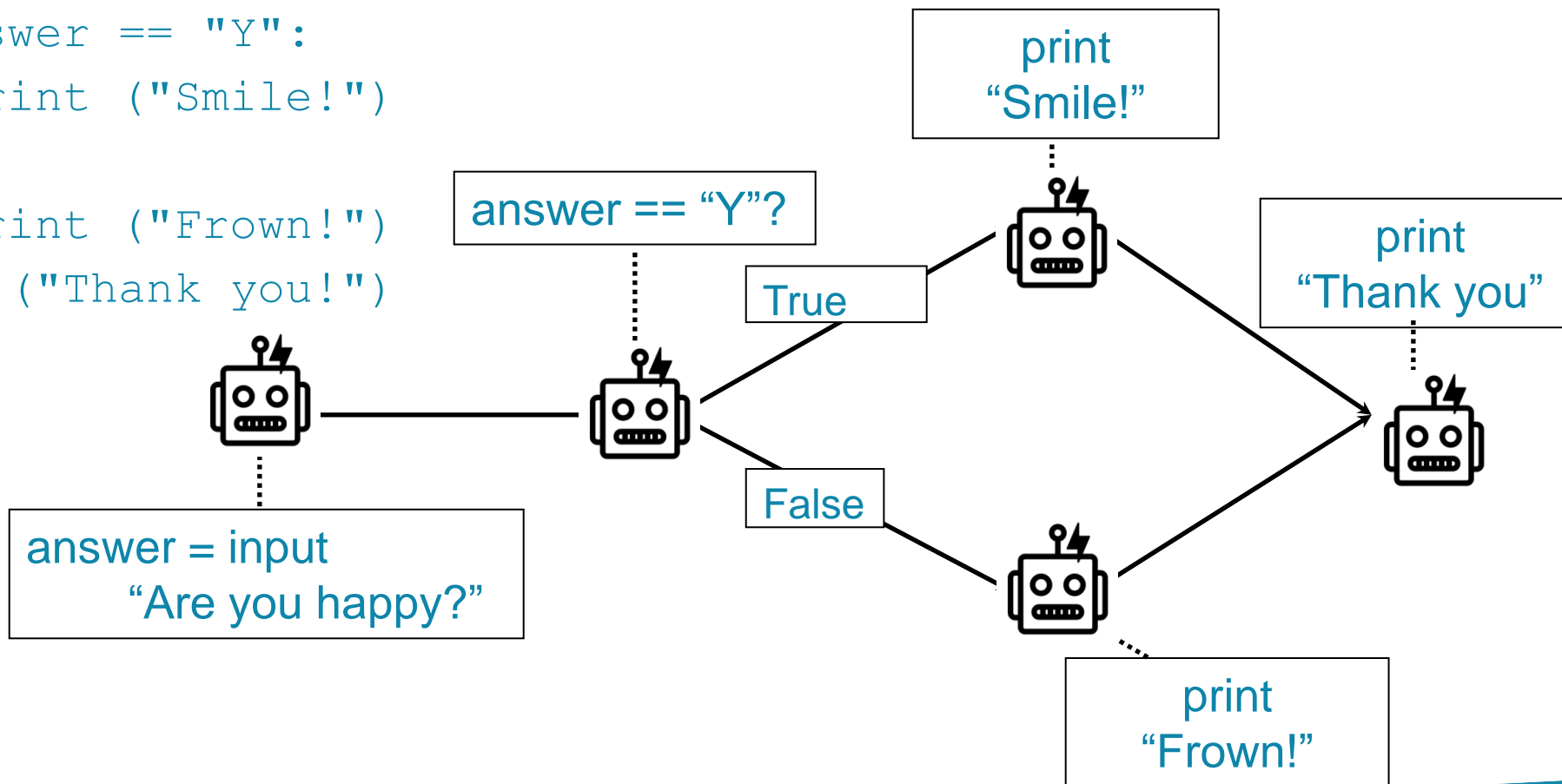
Group Activity





Role play

```
answer = input ("Are you happy?")  
if answer == "Y":  
    print ("Smile!")  
else:  
    print ("Frown!")  
print ("Thank you!")
```





Successful Pedagogies

✓ Analogy/storytelling

✓ CS Unplugged

Kinaesthetic

Role-playing

Puzzles

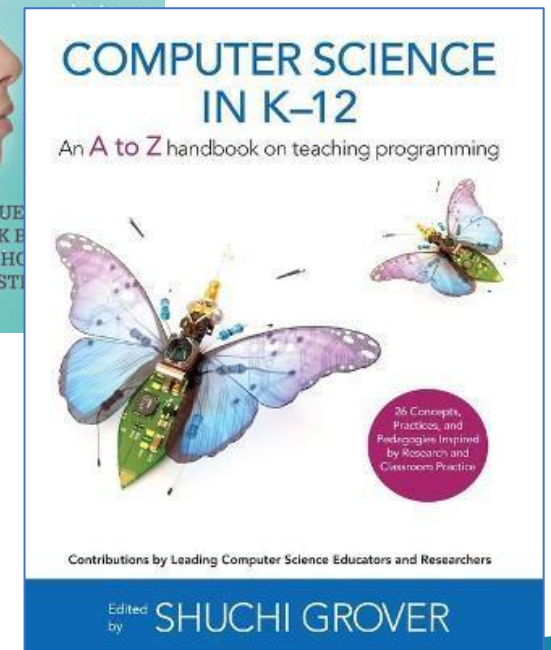
Art

Games

Magic

✓ Enquiry Based Learning (TEMI)

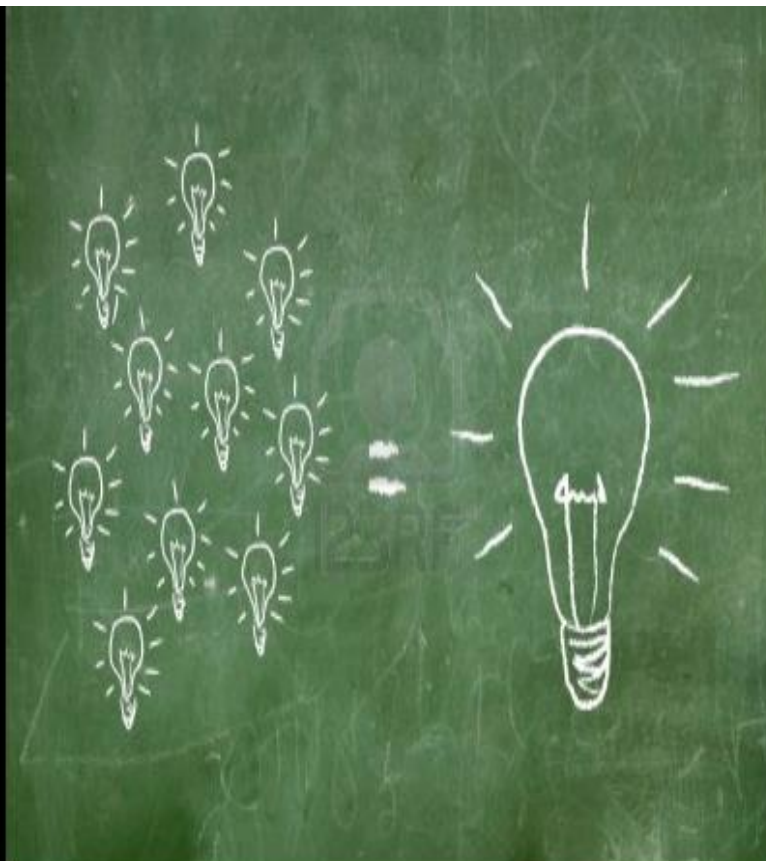
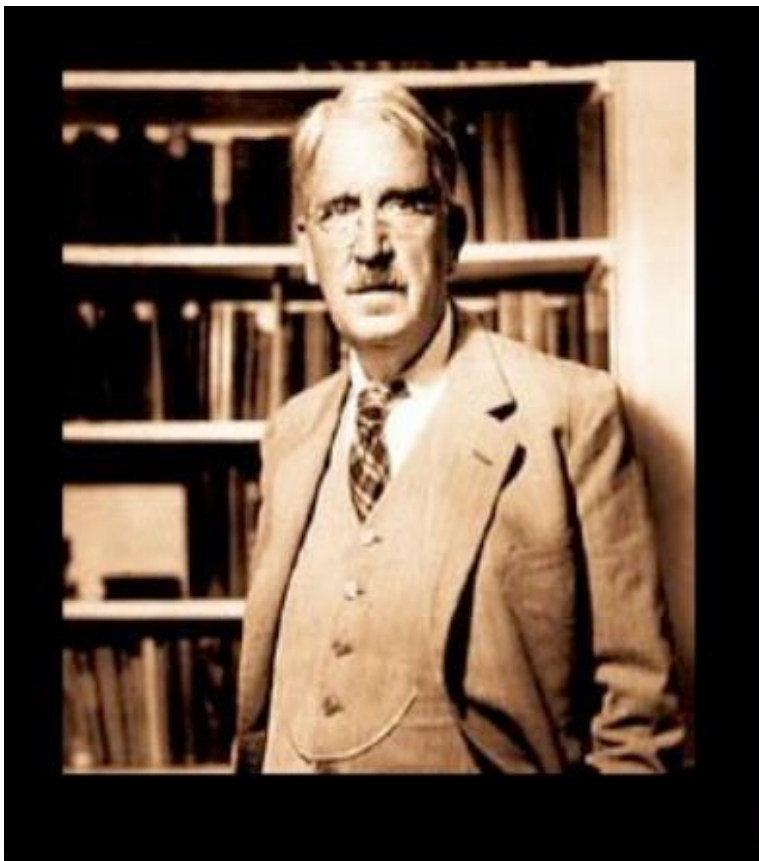
Programming Practice (Python / JavaScript)






Conclusion

- Computational Thinking tasks can be presented in a variety of ways
- Classroom context is key to deciding how to engage with certain topics
- There is no one way to think computationally (e.g. one's pattern recognition may be another's abstraction)
- The process of problem solving is non-linear and progress can be slow
- “Be less helpful”
- Confusion can be a good thing



 We only **THINK** when we are confronted with a **PROBLEM!**
John Dewey



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

NW7 Session 2: Formative Assessment for LCCS



LEAVING CERTIFICATE
COMPUTER SCIENCE





Overview of the session

Part 1	Significance of Assessment
Part 2	Formative Assessment Principles
Part 3	Formative Assessment using Digital Portfolios (FADP) initiative
Part 4	Digital Tools for Assessment in LCCS



By the end of this session participants will have:

- been given the opportunity to enhance their understanding of assessment and in particular formative assessment
- reflected on the importance of sharing learning intentions and success criteria and their own engagement with this
- reflected on the importance of effective feedback
- been introduced to some digital tools which can be used to support assessment and effective feedback in Computer Science



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 1: Significance of Assessment

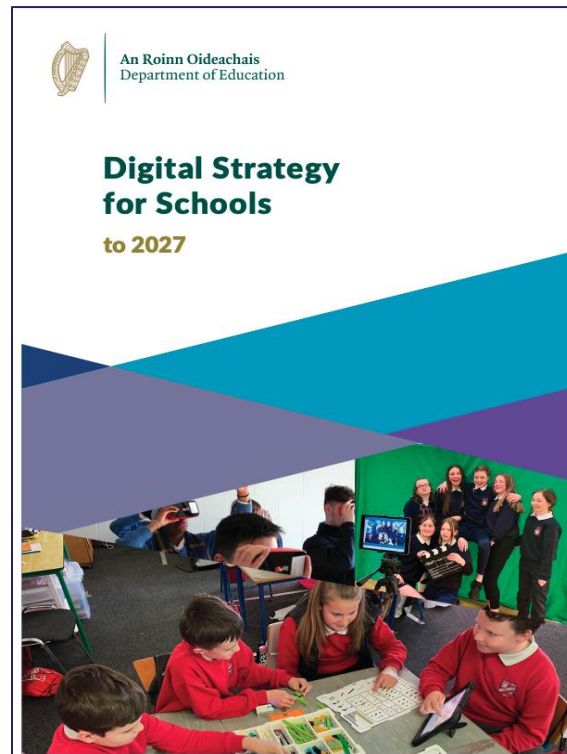


LEAVING CERTIFICATE
COMPUTER SCIENCE





Digital Strategy for Schools



*“Empower schools to harness the opportunities of digital transformation to build digital competence and an effective digital education ecosystem so as to develop **competent, critically engaged, active learners** while supporting them to reach their potential and participate fully as **global citizens in a digital world**”.*

Digital Strategy for Schools to 2027



Looking at Our School 2022



“The teacher selects and uses preparation and assessment practices that progress pupils’ learning”.

“Teachers collectively develop and implement consistent and dependable formative and summative assessment practices”.



Looking at Our School 2022



*“The principal, the deputy principal and other leaders in the school expect and encourage teachers to develop and extend their learning, teaching and assessment practices, and **to share and discuss practices that have proven successful at improving pupils’ learning**”.*

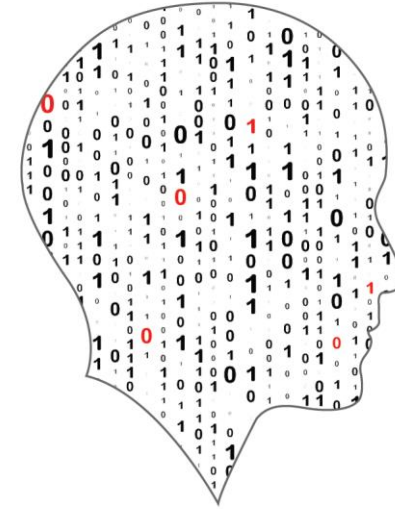


Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 2: Formative Assessment including effective feedback



LEAVING CERTIFICATE
COMPUTER SCIENCE





Defining assessment

- Assessment is the process of generating, gathering, recording, interpreting, using and reporting evidence of learning in individuals, groups and systems. Educational assessment provides information about progress in learning, and achievement in developing skills, knowledge, behaviours and attitudes.

(NCCA, 2015)

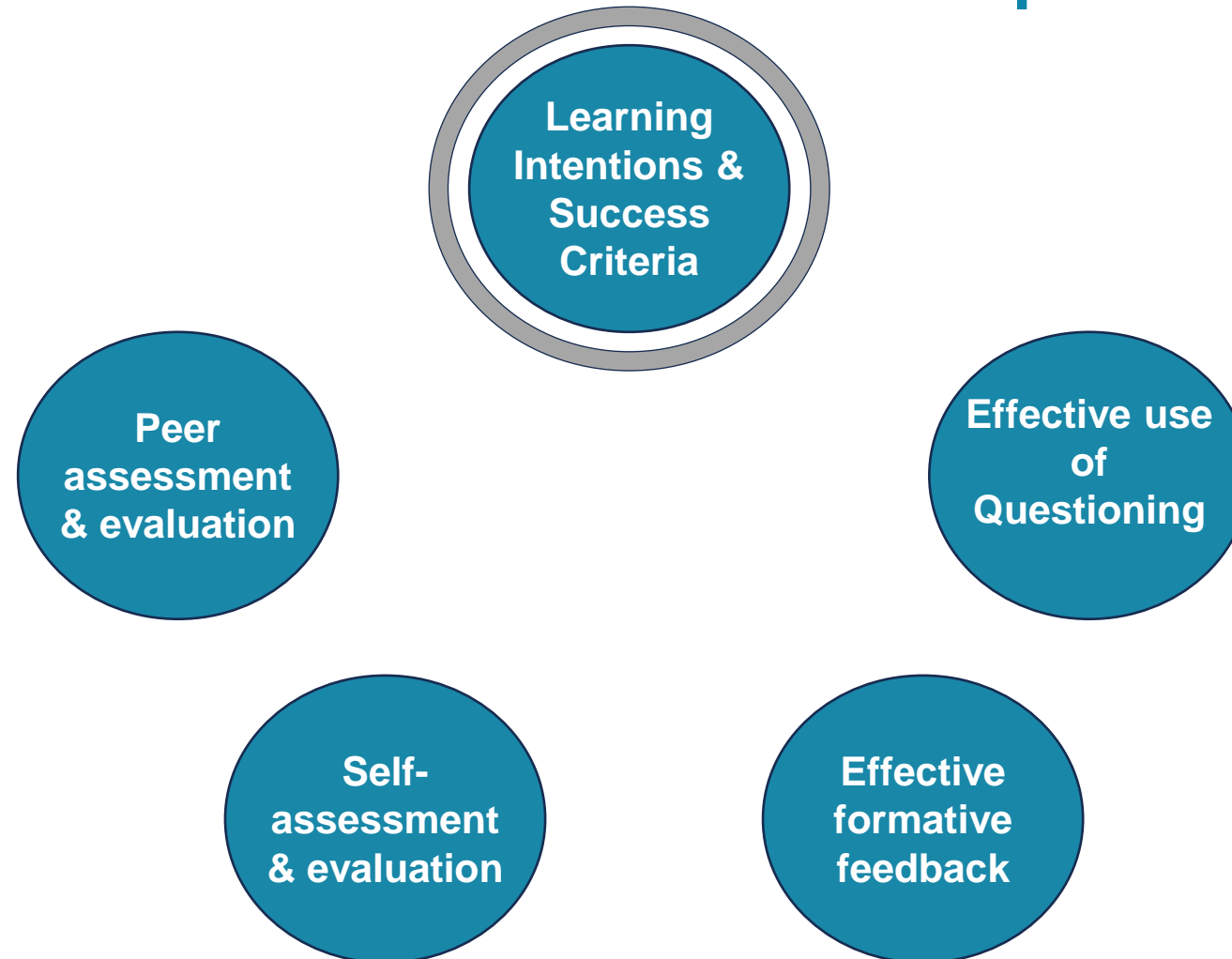


Formative Assessment





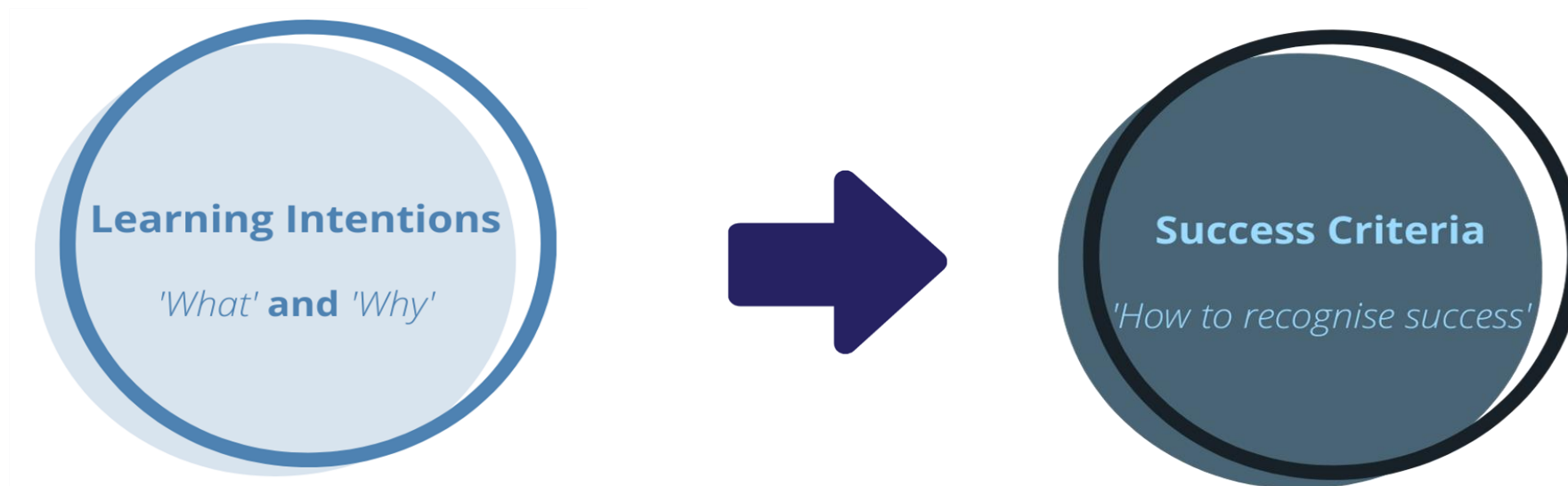
Formative Assessment Principles



William & Leahy (2015)
and Wylie et al (2008)



Learning Intentions and Success Criteria



An Introduction to AfL, Learning Unlimited (2004)



What is a Learning Intention?

- A statement, created by the teacher
- Describes clearly what the teacher wants the students to know, understand and be able to do
- Frequently linked to one or more learning outcome





Learning Intentions should...

Clear: focus on what will be learned in the lesson, as distinct from what students will do in the lesson

Useful: focus is on concepts, skills or knowledge that is used rather than focusing on imparting knowledge

Transferable to a similar context

(Adapted from source: Leahy, S, Lyon, C and Wiliam, D. (Nov.2005) Classroom Assessment: Minute by Minute, Day by Day. Educational Leadership)



Example of a Learning Intention

**LCA Information and
Communication Technology**

Module: The Internet and Digital
Literacy

Unit: Principles and Practices

**Learning
Outcome**

Present
information in
online digital
formats suitable
for the required
audience.



**Learning
Intention**

We will be able to
create a
presentation to
explain a topic to
an audience of
1st year
students/novices.



What are Success Criteria?

- Linked to learning intentions
- Developed by the teacher and/or students
- Describe what success looks like





Success Criteria should...

- Strengthen student learning
- Encourage independent learning
- Inform students what is expected of them
- Enable effective (formative) feedback





Success Criteria could be...

- a series of steps/sequence of instructions
- a list of options/menu from which the students can choose
- a list of “remember to” prompts
- a visual aid
- a rubric



Learning Intention to Success Criteria

Learning Outcome

Present information in online digital formats suitable for the required audience.



Learning Intention

We will be able to create a presentation to explain a topic to an audience of 1st year students/novices.



Success Criteria

- Presentation should contain 6 slides, including the title slide
- The introduction should clearly outline the topic to be discussed
- Each slide should have 2 to 4 short bullet points along with a script that explains each point in greater detail
- There should be at least one opportunity for audience engagement
- Take copyright into consideration for any external content you use (text, images, video, etc.)



Personal Reflection

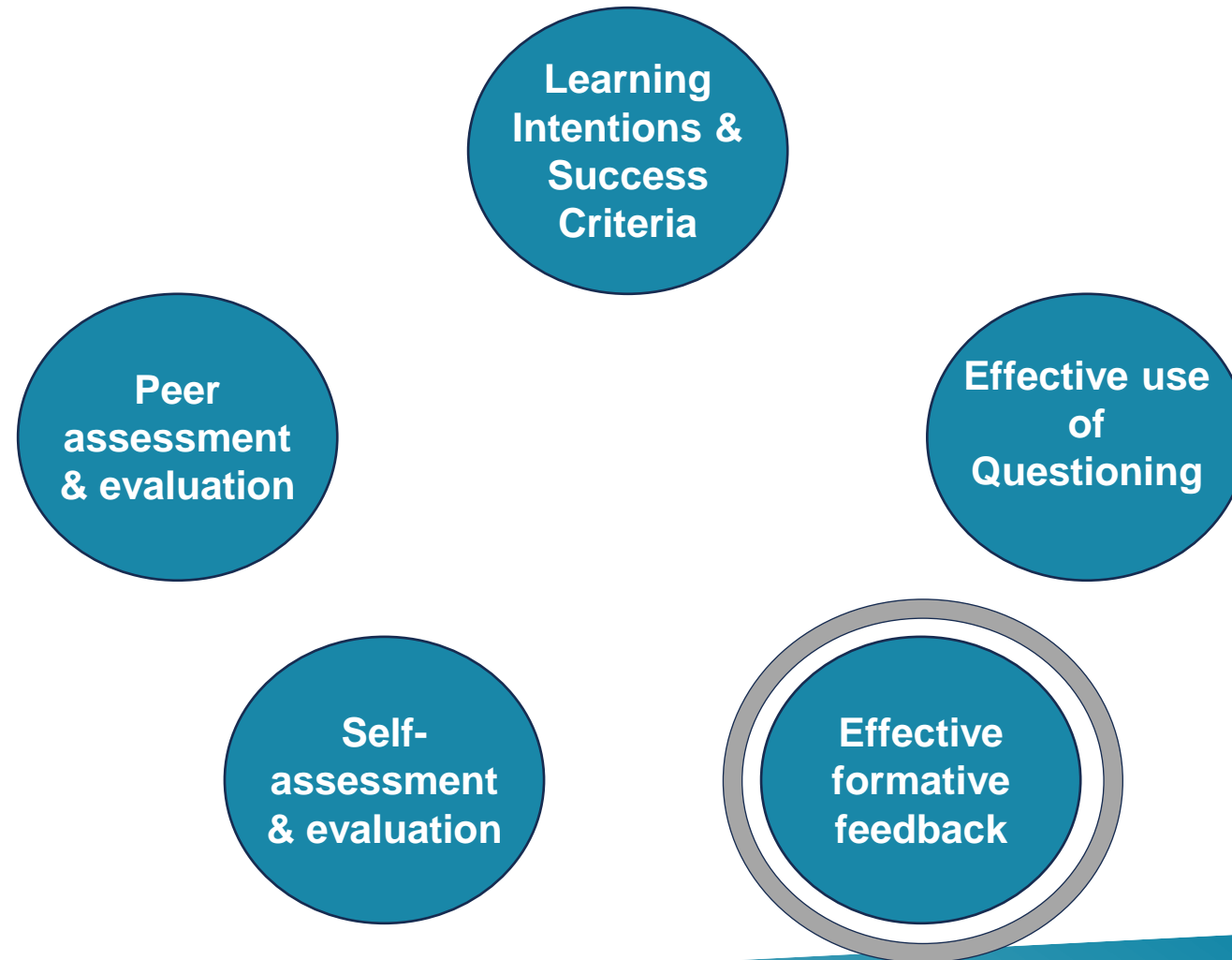


How often do I share learning intentions with my students?

How often do I share success criteria when I assign work to my students?



Formative Assessment Principles



William & Leahy (2015)
and Wylie et al (2008)



Formative Feedback

“

The most simple prescription for improving education must be dollops of feedback. This does not mean using many tests and providing over-prescriptive directions. It means providing information about how and why the student understands and misunderstands, and what directions the student must take to improve.

”

John Hattie, *Influences on Student Learning*



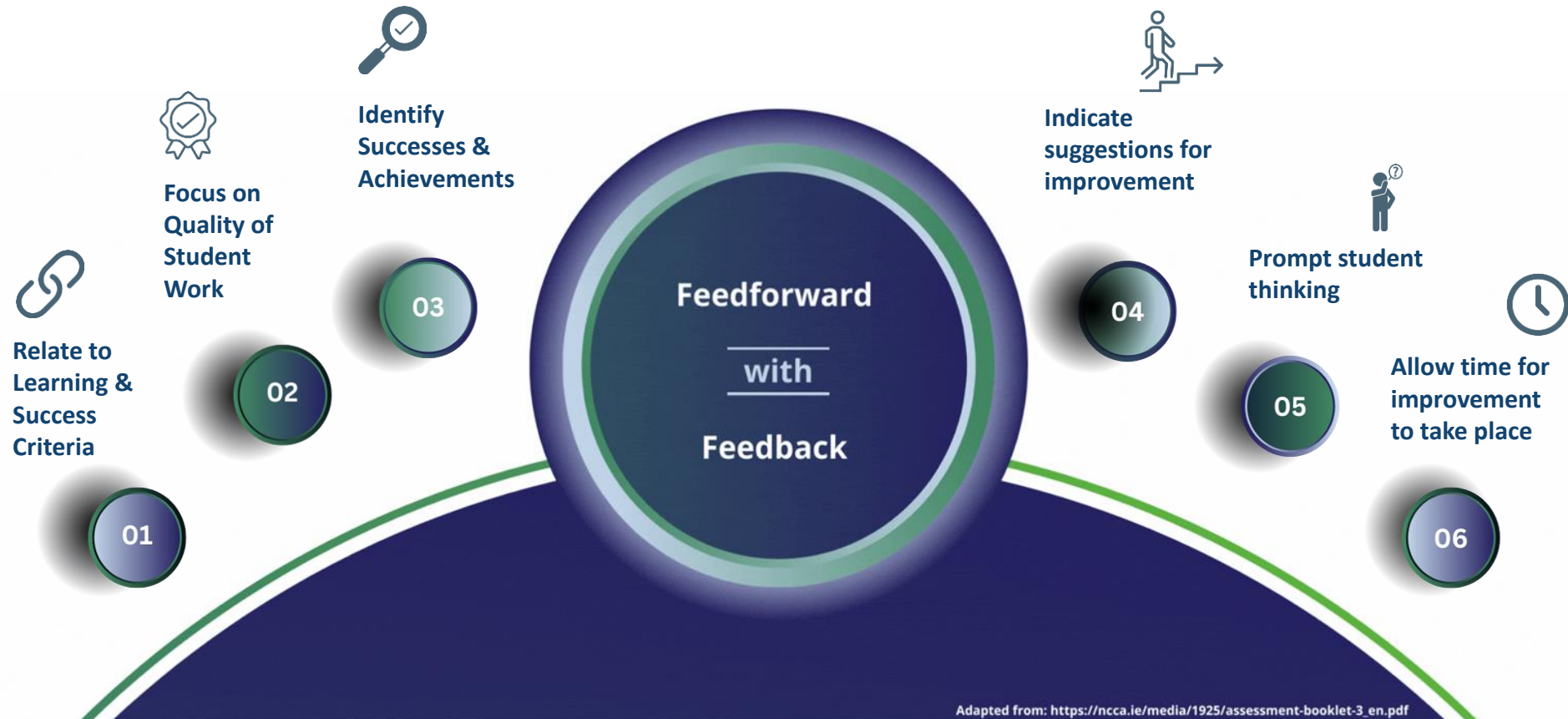
The Importance of Feedback on Learning



<https://youtu.be/n7Ox5aoZ4ww>



Effective Feedback





Personal Reflection



How often do I provide feedback to students that prompts thinking?

How often do I allow time for improvement to take place?



NCCA's Workshop Series



Sharing Learning
Intentions & Success
Criteria

Effective Questioning

Formative Feedback

Students Reflecting on
Learning

Learning Outcomes

<https://ncca.ie/en/junior-cycle/assessment-and-reporting/focus-on-learning/>



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 3: Formative Assessment using Digital Portfolios(FADP) initiative



LEAVING CERTIFICATE
COMPUTER SCIENCE





What are Digital Portfolios?

Defining Digital Portfolios

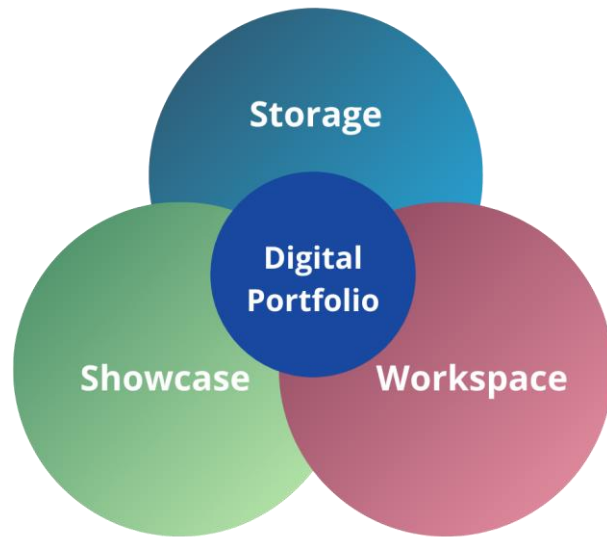
“ePortfolios are student-owned, dynamic digital workspaces whereby students can capture their learning, their ideas, access their collections of work, reflect on their learning, share it, set goals, seek feedback and showcase their learning and achievements.”







NCCA, 2013





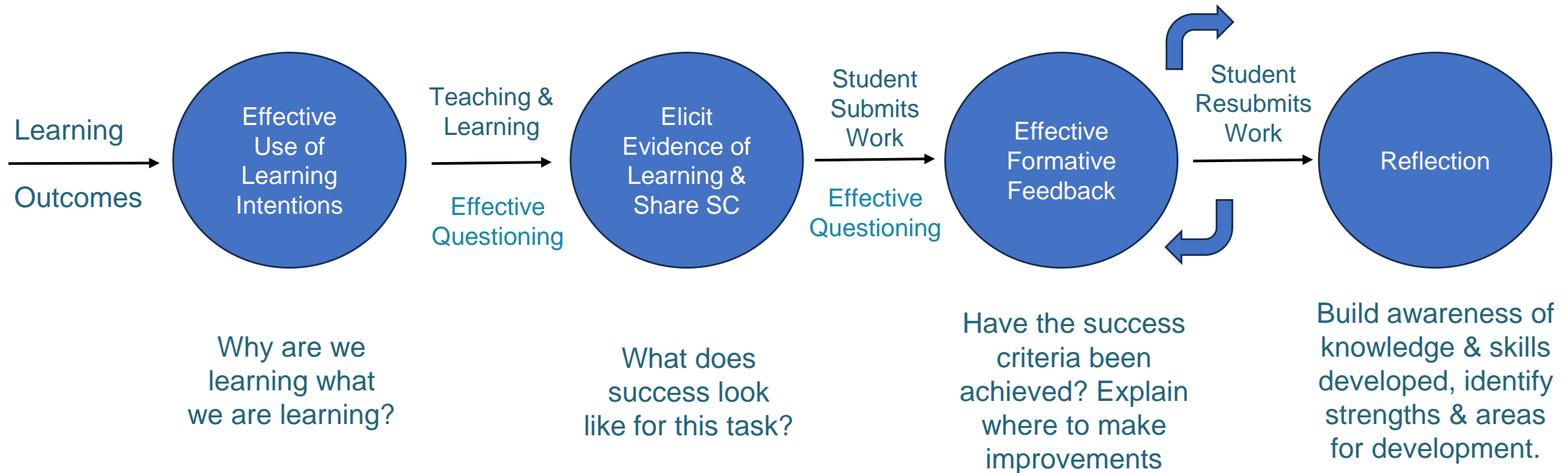
Elements of a Digital Portfolio



	Google	Office 365
Storage	Drive 	OneDrive 
Workspace	Classroom 	Teams/ OneNote 
Showcase	Sites 	Sway 



Formative Assessment using Digital Portfolios Process



Use Your School's Digital Platform to Enhance the Process & Develop Student's Digital Skills

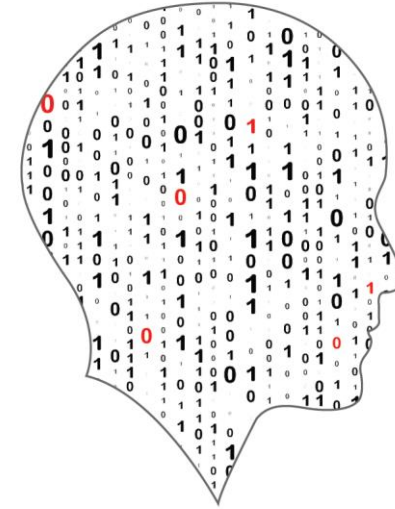


Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 4: Digital Tools for Assessment in LCCS

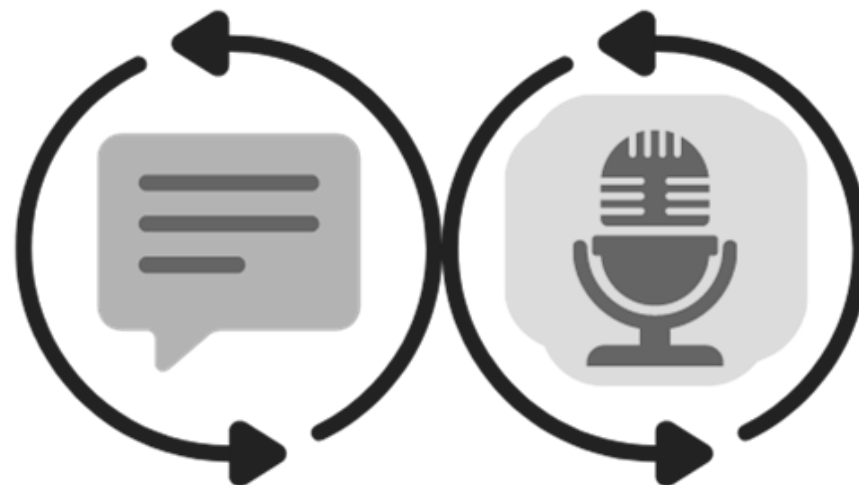


LEAVING CERTIFICATE
COMPUTER SCIENCE



Types of Formative Feedback

Written
Feedback



Oral
Feedback

- Quality V's Frequency
- Read and Respond
- Link to LI and SC
- Short

- Most effective
- Most natural
- Most frequent
- Evidence?



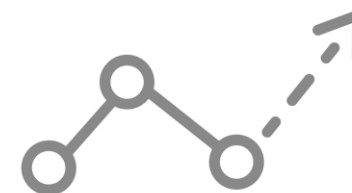
Benefits of Using Digital Tools for Feedback



Conducted remotely



Record of feedback



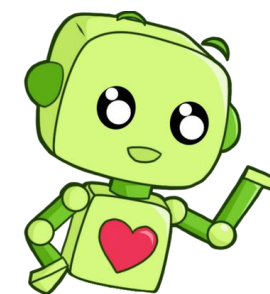
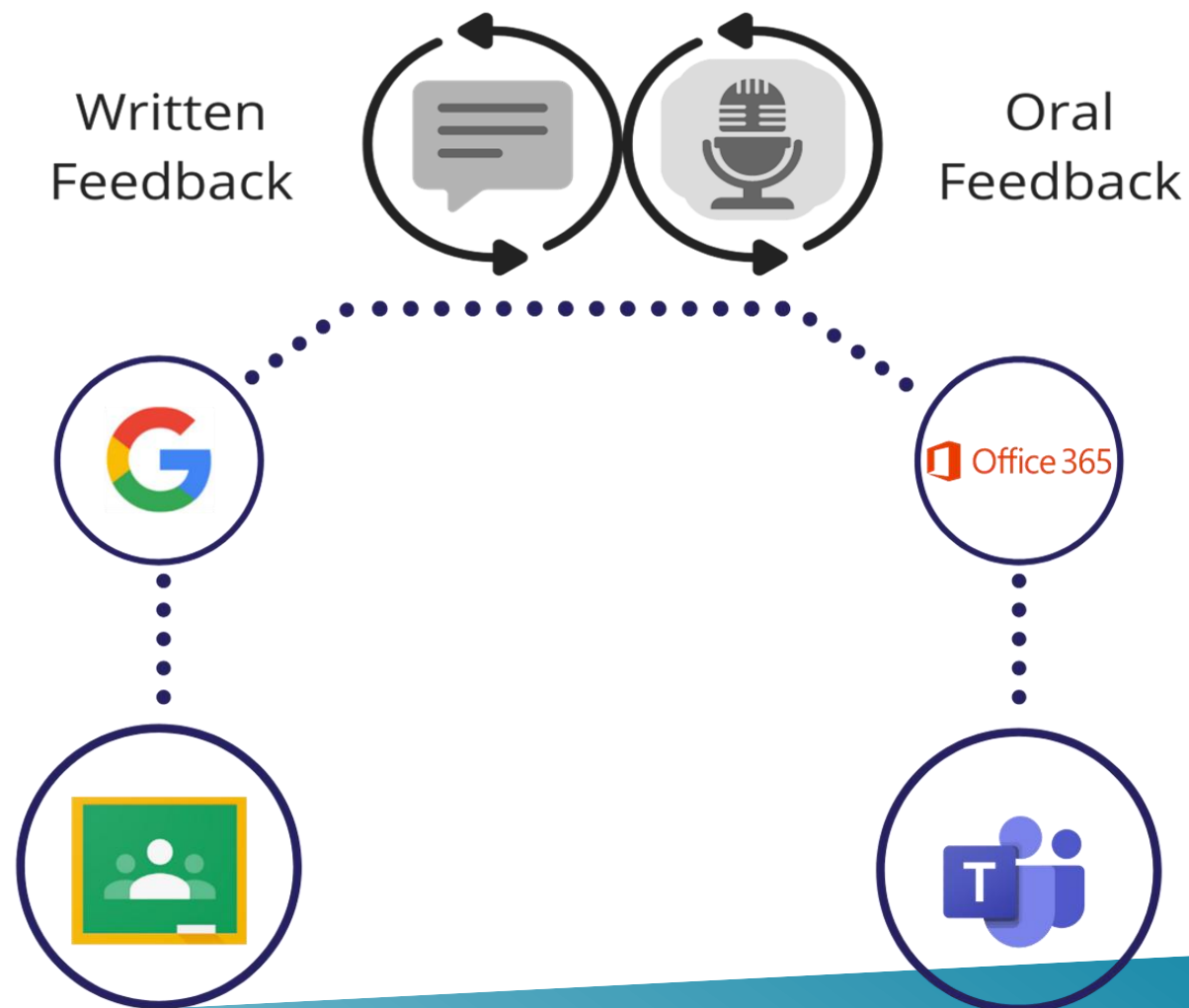
Student progress

“The principal and other leaders in the school lead a process of empowering teachers to embed digital technologies in their learning, teaching and assessment practices, and regularly evaluate the effectiveness of the use of these technologies”.

LAOS 2022



Digital Tools for Formative Feedback






Vocaroo





Home Expert Activity: Digital Tools for Assessment in the LCCS Classroom

Home Expert Review

1  edpuzzle	2 	
3  Diagnostic Questions	4  Google Forms/ Microsoft Forms	5  Kahoot!



1. Give a brief description of the application
2. List **three benefits** of the application in the LCCS classroom
3. Give one example of how you might use the application for **assessment** in the LCCS classroom



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Session 3: Computer Systems IV



LEAVING CERTIFICATE
COMPUTER SCIENCE





Computer Systems and the Specification

“The core concepts are developed theoretically and applied practically. In this way, conceptual classroom-based learning is intertwined with experimental computer lab-based learning throughout the two years of the course.” *NCCA Curriculum specification, Page 20*

Strand 1: Practices and principles	Strand 2: Core concepts	Strand 3: Computer science in practice
<ul style="list-style-type: none">▶ Computers and society▶ Computational thinking▶ Design and development	<ul style="list-style-type: none">▶ Abstraction▶ Algorithms▶ Computer systems▶ Data▶ Evaluation/Testing	<ul style="list-style-type: none">▶ Applied learning task 1<ul style="list-style-type: none">- Interactive information systems▶ Applied learning task 2 - Analytics▶ Applied learning task 3<ul style="list-style-type: none">- Modelling and simulation▶ Applied learning task 4<ul style="list-style-type: none">- Embedded systems



LCCS Learning Outcomes

Strand 2: Abstraction

- 2.1 use abstraction to describe systems and to explain the relationship between wholes and parts
- 2.2 use a range of methods for identifying patterns and abstract common features
- 2.3 implement modular design to develop hardware or software modules that perform a specific function
- 2.4 illustrate examples of abstract models

Strand 2: Computer Systems

CPU: ALU, Registers, Program counter, Memory

Operating system layers: Hardware, OS, Application, User

- 2.11 describe the different components within a computer and the function of those components
- 2.12 describe the different types of logic gates and explain how they can be arranged into larger units to perform more complex tasks

Computer Network Protocols: HTTP, TCP, IP, VOIP

- 2.15 explain what is meant by the World Wide Web (WWW) and the Internet, including the client server model, hardware components **and communication protocols**



Layers of a Computing System



Oide

End-user

Application Software

System Software

Computer Components

Electronics, Gates and Circuits

Data Representation



Group Activity



Data Representation



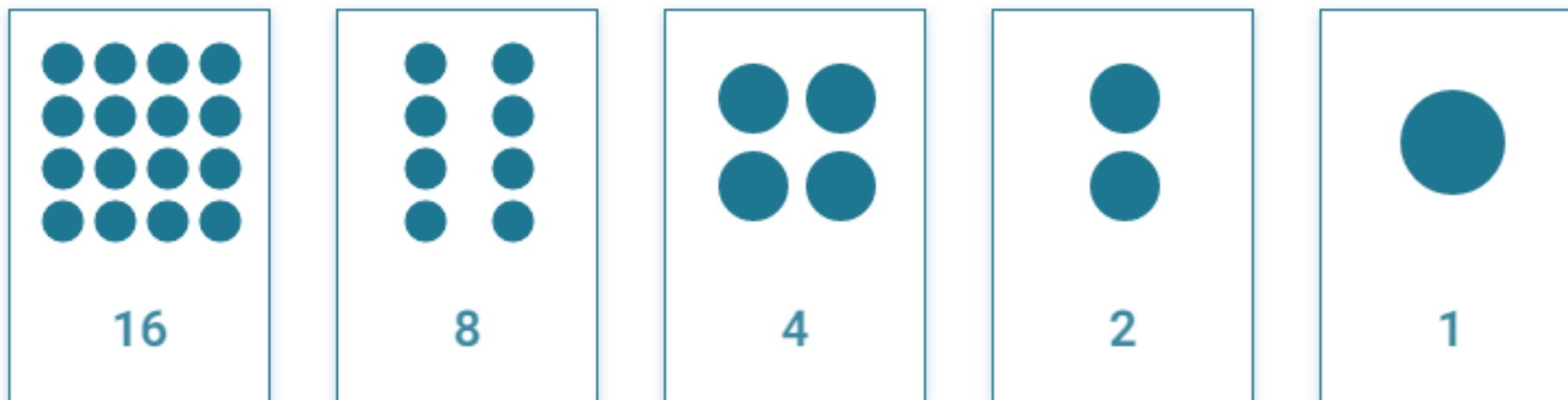
Oide



<https://youtu.be/b6vHZ95XDwU>



Data Representation



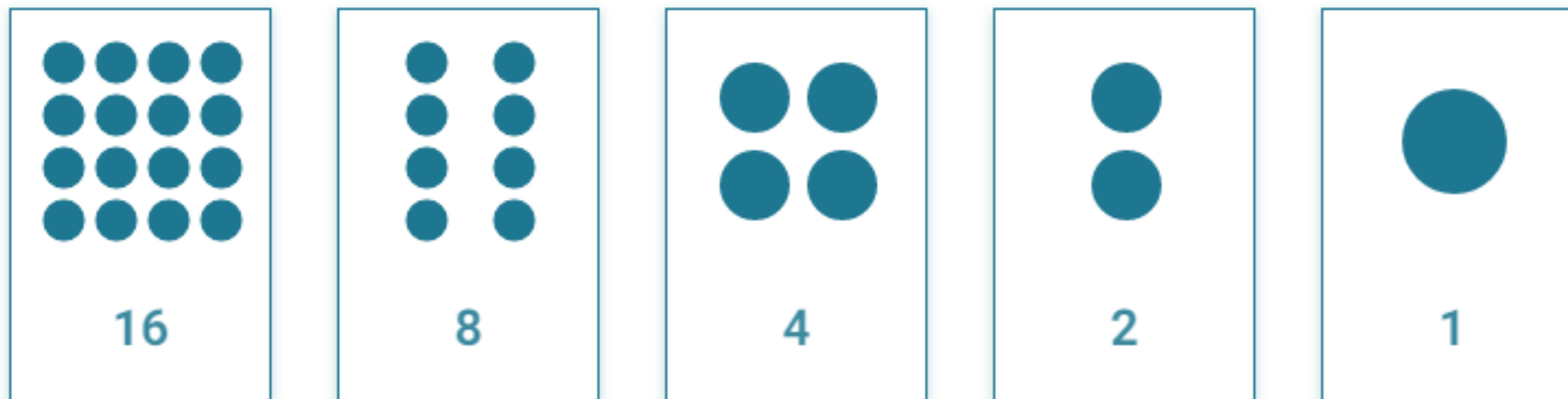
Which cards do we need to turn over to make the number 13?

(The cards are black on the reverse side.)

Data Representation



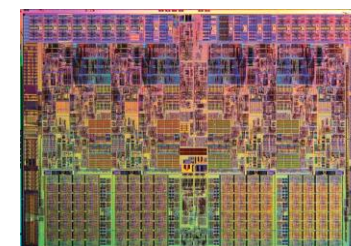
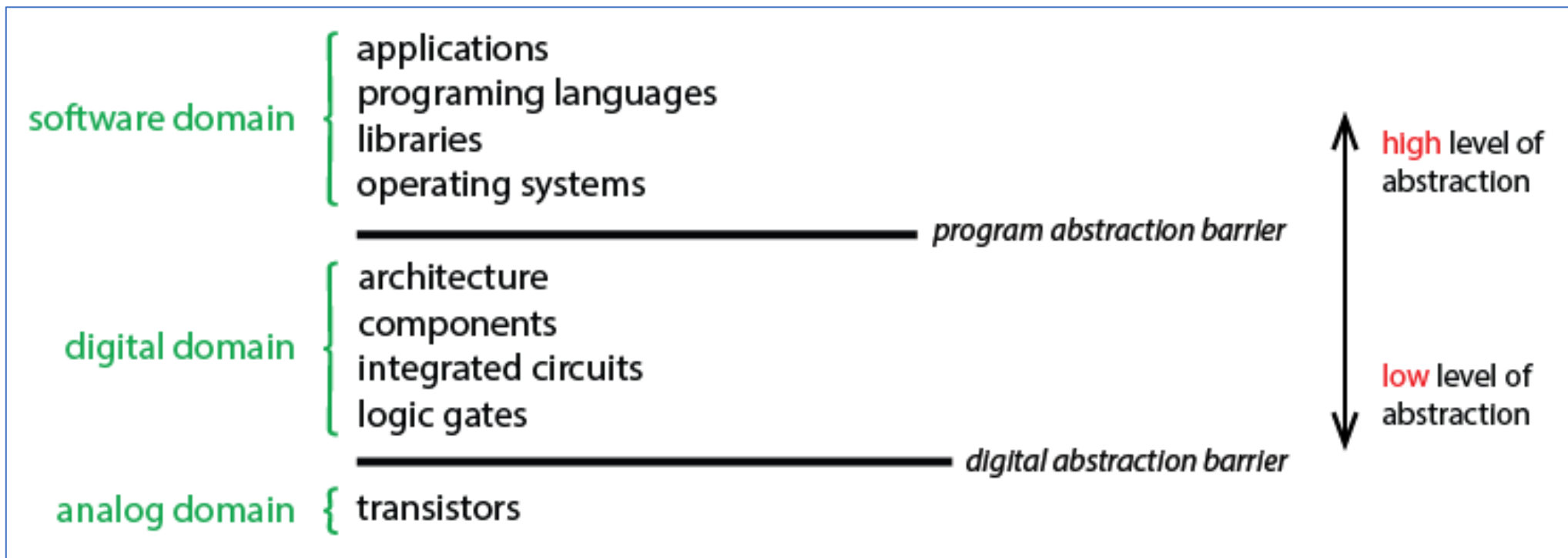
Data Representation



What is the biggest number that can be represented using the above cards?
What is the smallest number that can be represented using the above cards?
How can we represent a bigger number?



Abstraction inside the computer





Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Computer Components



LEAVING CERTIFICATE
COMPUTER SCIENCE





What Makes a Computer, a Computer?



<https://youtu.be/mCq8-xTH7jA>



Von Neumann Architecture

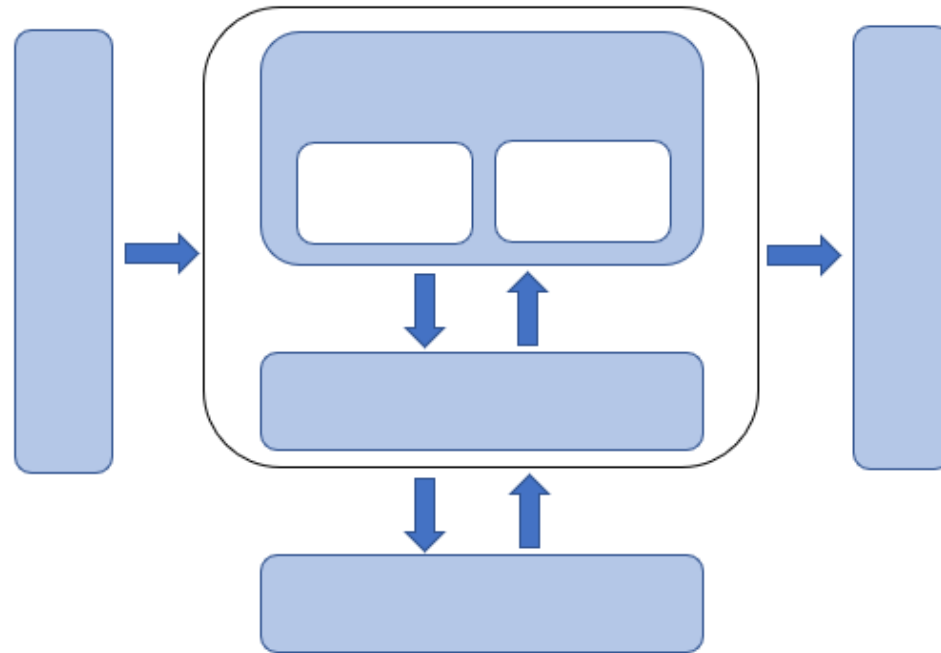


<https://youtu.be/MI3-kVYLNr8>



Activity: Von Neumann Architecture

2.11 describe the different components within a computer and the function of those components



The components are connected to one another by a collection of wires called a bus

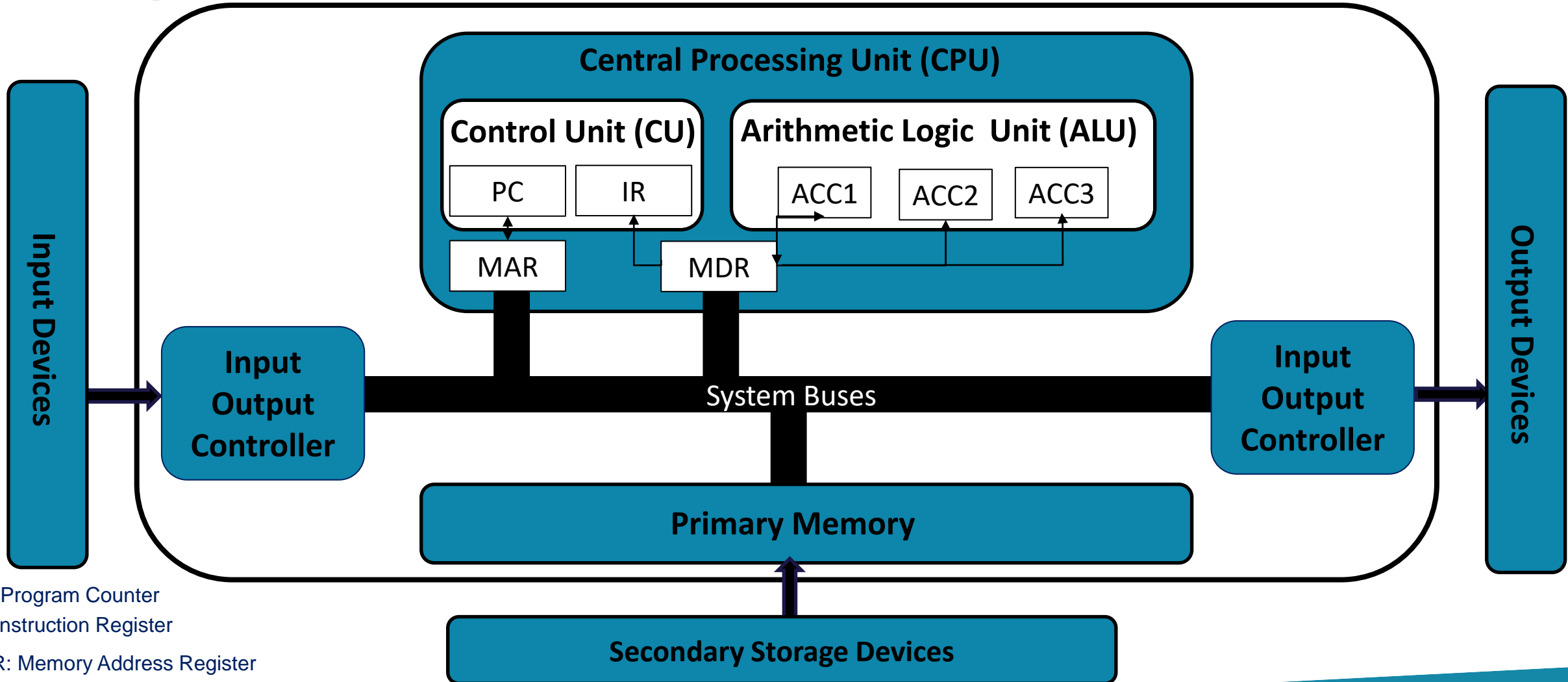
Computer Components



Registers

- Storage locations internal to the CPU
- Used as a scratchpad by the CPU to store data, addresses or instructions as it executes each program instruction
- Data can be moved into and out of registers faster than from memory – dedicated pathways and hardware

The von Neumann Architecture of a Computer



PC: Program Counter

IR: Instruction Register

MAR: Memory Address Register

MDR: Memory Data Register

ACC: Accumulator



The Fetch-Execute Cycle

At the start of the fetch-execute cycle, the Program Counter contains the address of the next instruction to be executed

Step 1. Fetch

The contents of the address in the PC are loaded from memory to the Instruction Register (IR) (via the MAR and MDR)

Step 2. Decode

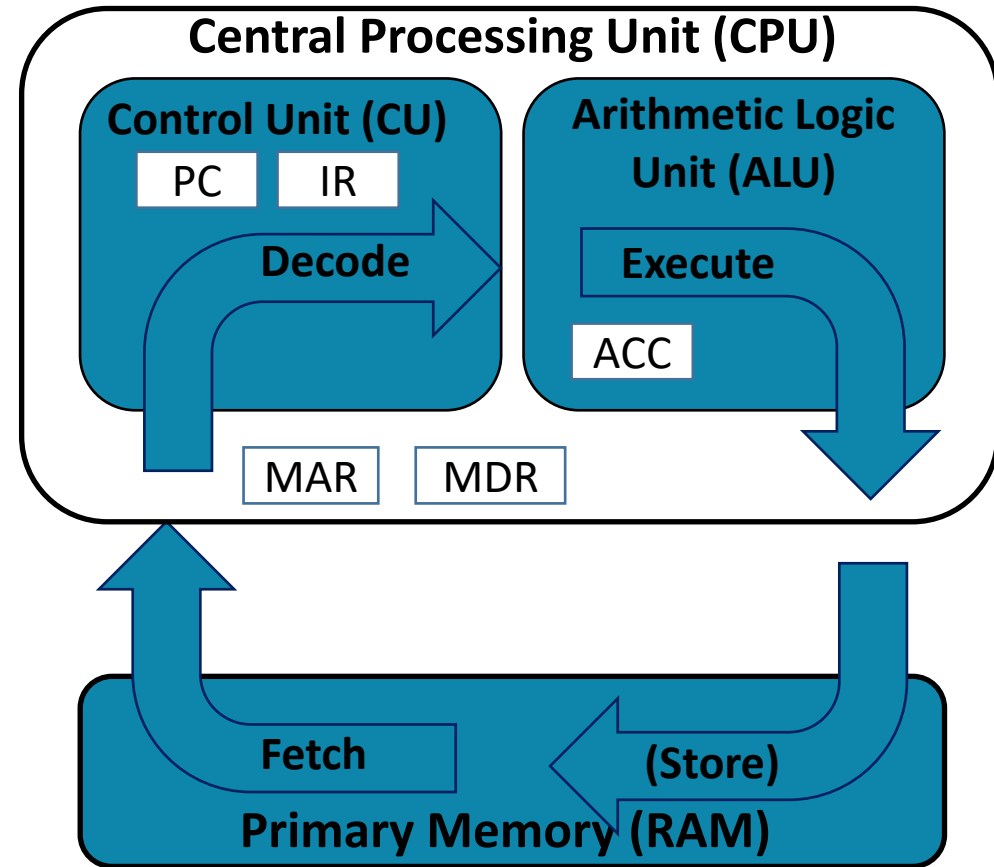
The control unit determines the type of instruction to be carried out and sets the control signals accordingly.

Step 3. Execute

The operation is carried out by the CU or the ALU (depending on the instruction – load and store operations are carried out by the CU while arithmetic and logical operations are carried out in the ALU).

Step 4. Store

The result of the operation is written to registers or RAM



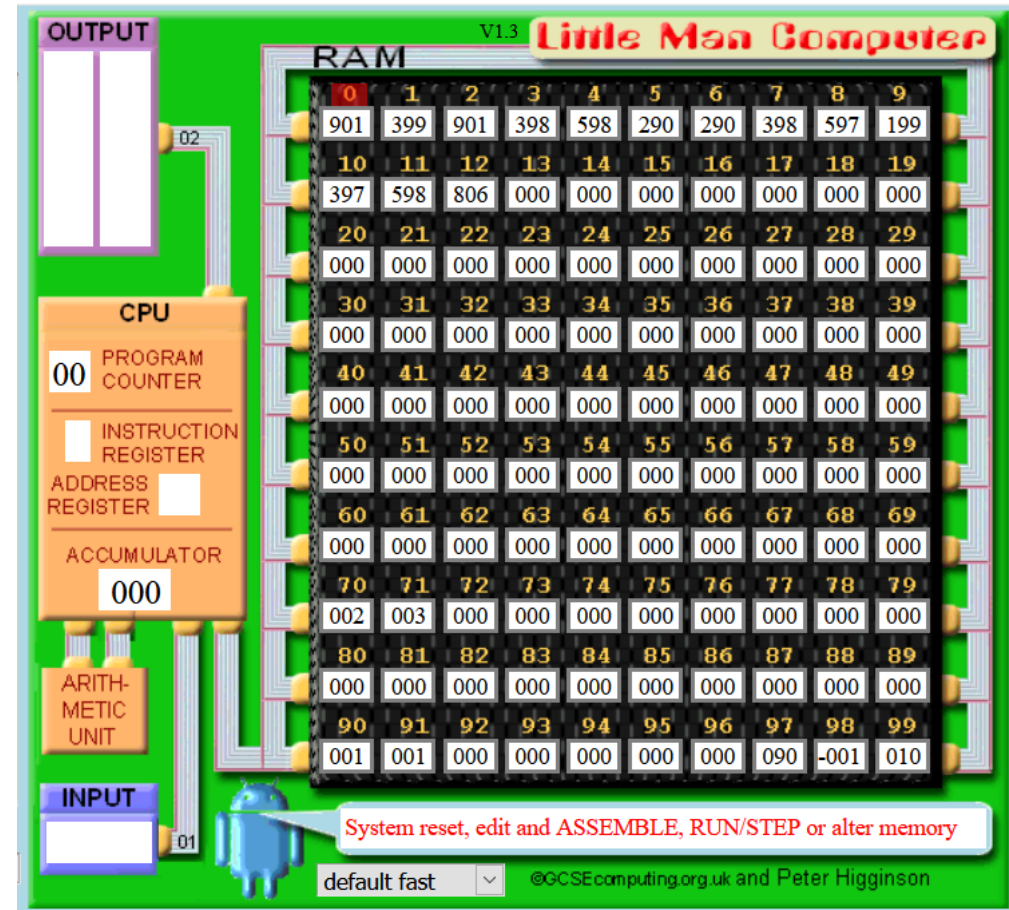
<https://www.youtube.com/watch?v=jFDMZpkUWCw>

<https://www.youtube.com/watch?v=04UGopESS6A>



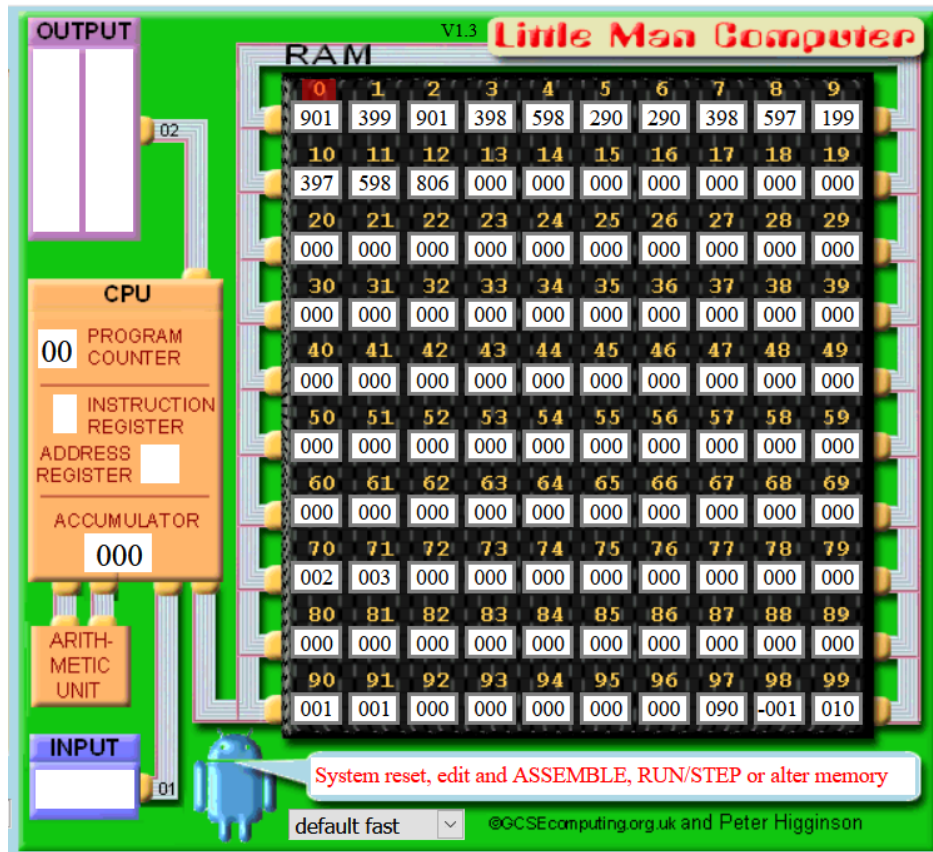
Little Man Computer

Little Man Computer Simulator:
<https://peterhigginson.co.uk/LMC/>





Little Man Computer



Little Man Computer Demo:
<https://www.futurelearn.com/info/courses/how-computers-work/0/steps/49285>

Little Man Computer Simulator:
<https://peterhigginson.co.uk/LMC/>

Little Man Computer Help:
<https://peterhigginson.co.uk/LMC/help.html>

Fetch Execute Decode Cycle:
<https://www.futurelearn.com/info/courses/how-computers-work/0/steps/49284>

(Another) Little Man Computer Simulator:
<https://www.101computing.net/lmc-simulator/>



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Operating Systems



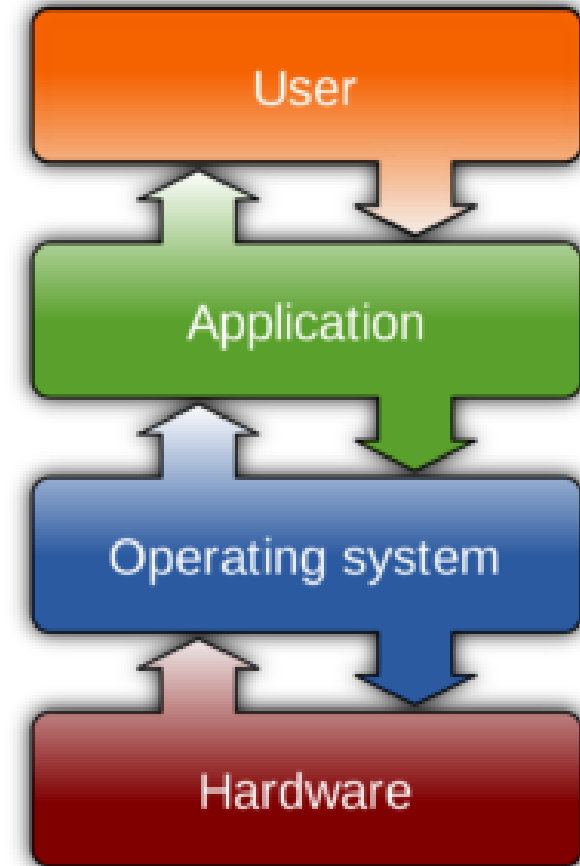
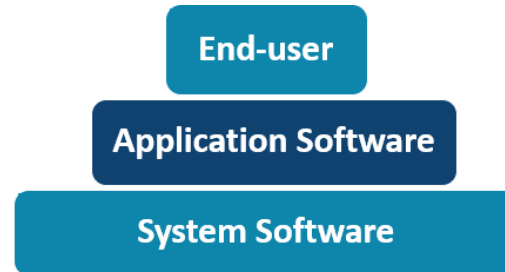
LEAVING CERTIFICATE
COMPUTER SCIENCE





Layers of an OS

- Hardware
- Operating System
- Application
- User



	using any algorithms studied
S2: Computer systems	
CPU: ALU, Registers, Program counter, Memory	2.11 describe the different components within a computer and the function of those components
Basic electronics: voltage, current, resistors, capacitors, transistors	2.12 describe the different types of logic gates and explain how they can be arranged into larger units to perform more complex tasks
Operating system layers: Hardware, OS, Application, User	2.13 describe the rationale for using the binary number system in digital computing and how to convert between binary, hexadecimal and decimal
	2.14 describe the difference between digital and analogue input
Web infrastructure - Computer Network Protocols: HTTP, TCP, IP, VoIP	2.15 explain what is meant by the World Wide Web (WWW) and the Internet, including the client server model, hardware components and communication protocols

Functions of an Operating System

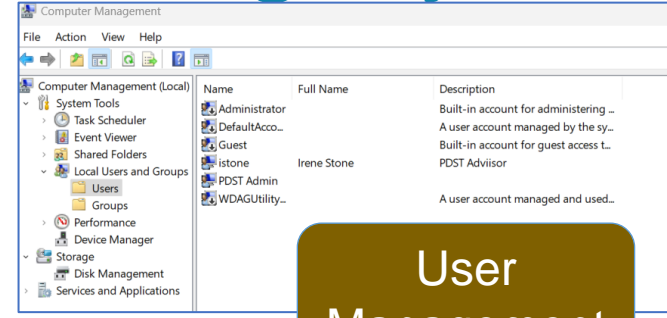


Oide



Process Management

Security Management



User Management



Mobile User Interface (MobUI)

Operating System

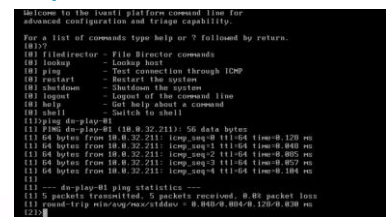
User Interface



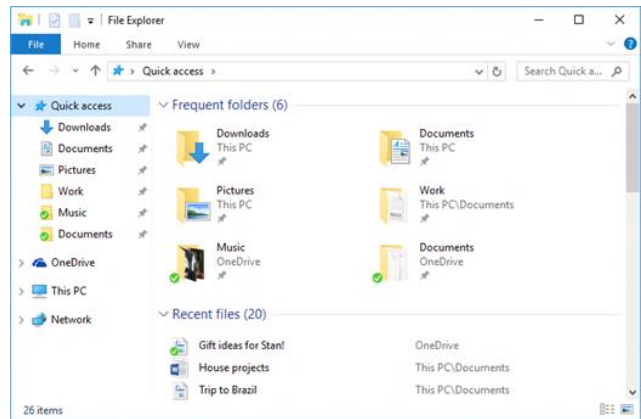
Graphical User Interface (GUI)

File Management

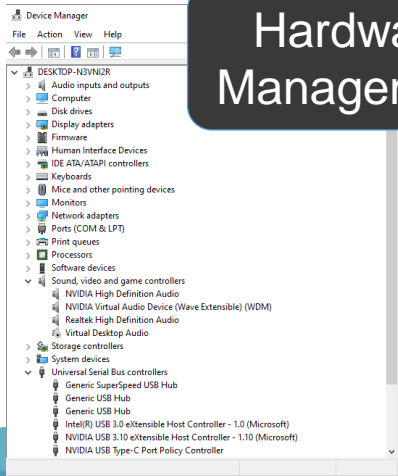
Memory Management



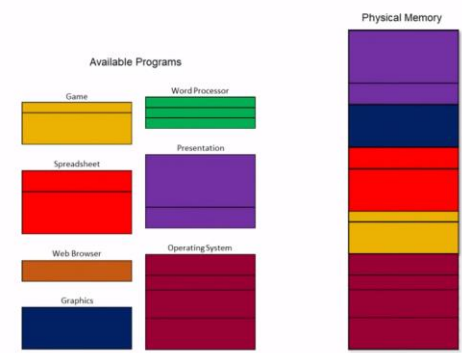
Command Line Interface (CLI)



Hardware Management

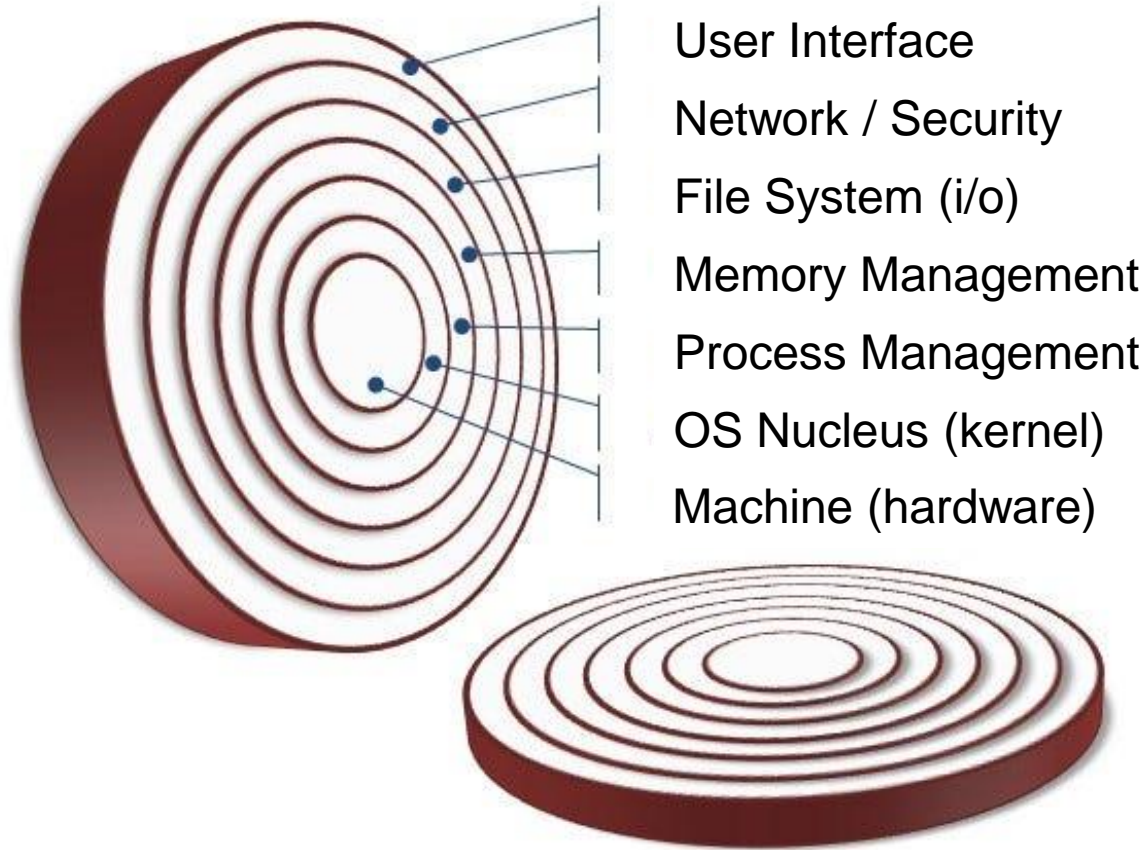


Segmented Memory





Operating System



- An operating system (OS) is a program that acts as an interface between the system hardware and the user.
- It handles all the interactions between the software and the hardware.
- All the working of a computer system depends on the OS at the base level.
- It performs all the functions like handling memory, processes, the interaction between hardware and software.



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Programming Languages

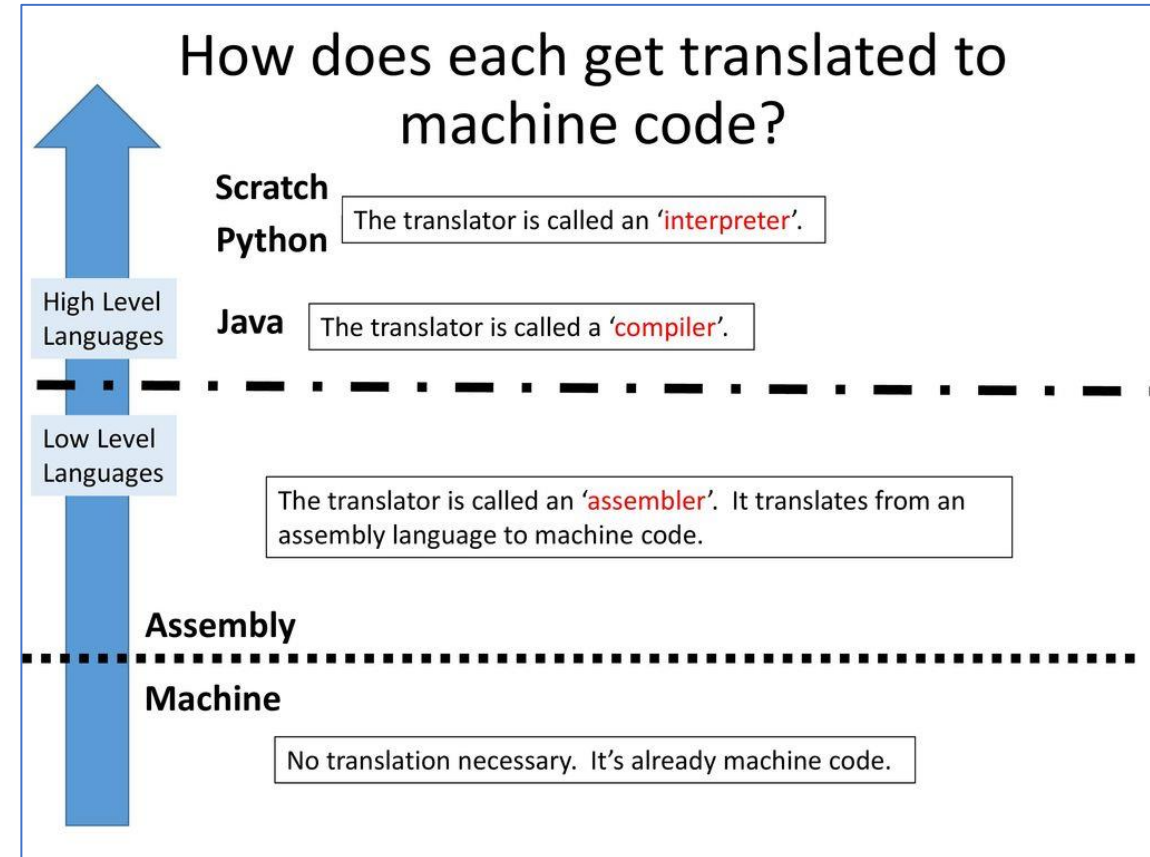
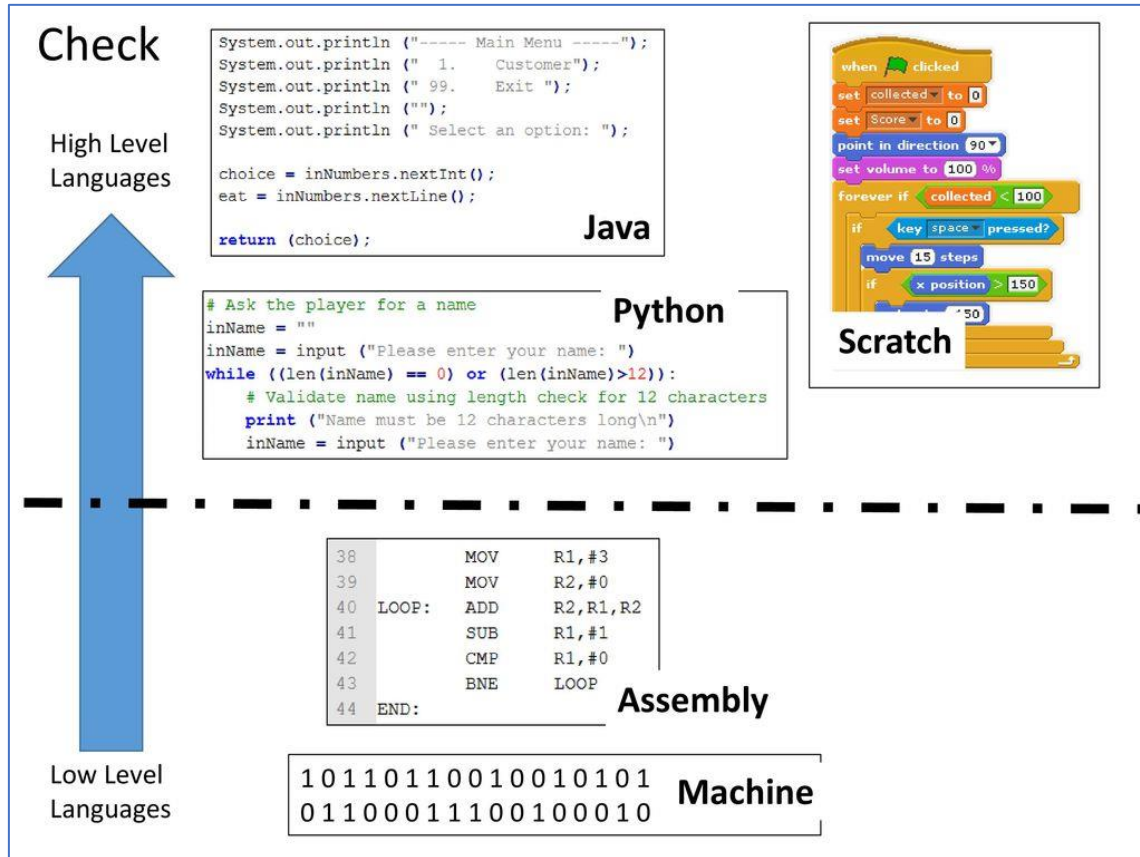


LEAVING CERTIFICATE
COMPUTER SCIENCE





High-level vs Low-level





Artificial Intelligence

"You may ask whether a child who uses a chatbot to generate a program is still actually doing programming. I would argue that the child is still programming — only they are doing so in English (or another human language a chatbot is trained to generate). They are programming at a significantly higher level of abstraction compared to using traditional programming languages"

Ken Kahn, University of Oxford





Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Computer Network Protocols



LEAVING CERTIFICATE
COMPUTER SCIENCE





Protocols

- All methods of communication need rules in place in order to pass on the message successfully. These sets of rules are called **protocols**.
- *What protocol exists for meeting someone new? Is this the same in all countries?*
- *In which cultures do people use the below protocols to greet each other?*
 - *Bowing*
 - *Rubbing noses*
 - *Sticking your tongue out*





Computer protocols

- *What rules (protocols) exist for an email address?*
- *What parts of this web address show rules (protocols) being used?*
- <https://www.oide.ie/>



Communication protocols

- Internet Protocols are a set of rules that govern the communication and exchange of data over the Internet. Both the sender and receiver should follow the same protocols in order to communicate the data.
- TCP/IP - Transmission Control Protocol/Internet Protocol - enables communication over the internet.
- HTTP and HTTPS - Hypertext Transfer Protocol - governs communication between a webserver and a client. HTTPS (secure) includes secure encryption to allow transactions to be made over the internet.
- FTP - File Transfer Protocol - governs the transmission of files across a network and the internet.




Internet Protocols

OCR	COMPUTER SCIENCE
GCSE	

J277

1.3
**COMMON
PROTOCOLS**



<https://youtu.be/ncGIs1Wnxn8>



Communication protocols

In your groups:

- Discuss how you have approached (or will approach!) this area of the course in your LCCS classroom.
- Prepare some feedback for the wider group.





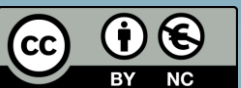
Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Leaving Certificate Computer Science National Workshop 7

Day 2





Workshop Overview

Session 1 09:00 - 11:00	Algorithms IV
Tea/Coffee 11:00 – 11:30	
Session 2 11:30 - 13:00	Computers in Society: Turing Machines
Lunch 13:00 - 14:00	
Session 3 14:00 - 15:30	Curriculum Planning



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Session 4: Algorithms IV



LEAVING CERTIFICATE
COMPUTER SCIENCE





Overview of the session

Part 1	Recap on Algorithms
Part 2	Algorithmic Complexity
Part 3	Analysis of Searching and Sorting Algorithms
Part 4	Limits of Algorithms & Heuristics



By the end of this session participants will have:

- reflected on their understanding of algorithms
- explained, through activities, the operation of a variety of searching and sorting algorithms.
- used an analysis framework to explore the time complexity of the aforementioned algorithms and in doing so deepened their understanding of these algorithms.
- reflected on ideas to facilitate the effective learning of algorithms in their own classrooms.

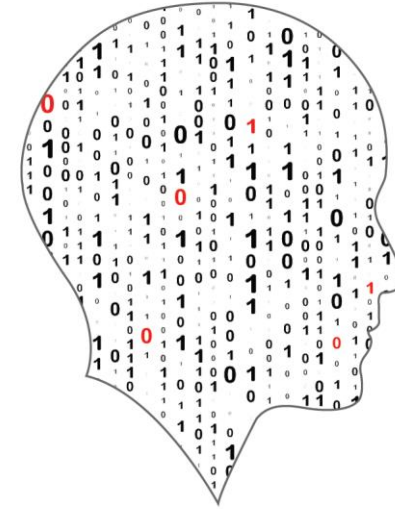


Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 1: Recap on Algorithms



LEAVING CERTIFICATE
COMPUTER SCIENCE





Algorithms and the Specification

“The core concepts are developed theoretically and applied practically. In this way, conceptual classroom-based learning is intertwined with experimental computer lab-based learning throughout the two years of the course.” *NCCA Curriculum specification, Page 20*

Strand 1: Practices and principles	Strand 2: Core concepts	Strand 3: Computer science in practice
<ul style="list-style-type: none">▶ Computers and society▶ Computational thinking▶ Design and development	<ul style="list-style-type: none">▶ Abstraction▶ Algorithms▶ Computer systems▶ Data▶ Evaluation/Testing	<ul style="list-style-type: none">▶ Applied learning task 1<ul style="list-style-type: none">- Interactive information systems▶ Applied learning task 2 - Analytics▶ Applied learning task 3<ul style="list-style-type: none">- Modelling and simulation▶ Applied learning task 4<ul style="list-style-type: none">- Embedded systems



LCCS Learning Outcomes

2.5 use pseudo code to outline the functionality of an algorithm

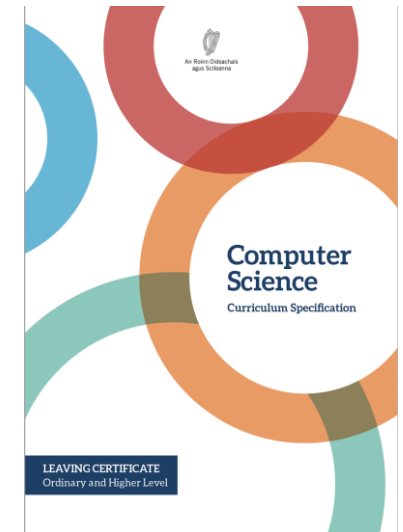
2.6 construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement

2.7 implement algorithms using a programming language to solve a range of problems

2.8 apply basic search and sorting algorithms and describe the limitations and advantages of each algorithm

2.9 assemble existing algorithms or create new ones that use functions (including recursive), procedures, and modules

2.10 explain the common measures of algorithmic efficiency using any algorithms studied





What is an algorithm?

A step-by-step procedure for solving a problem or accomplishing some end especially by a computer
Merriam-Webster

- A sequence of instructions
- A way of capturing intelligence and sharing it with others
- Provide general solutions to problems
- Some problems are so hard that they cannot be solved by algorithms (Computability)
- Can be expressed in a variety of different ways
- Common elements include input, processing, output
- Close relationship between algorithms and data structures
- Essential features are correctness and effectiveness
- Rule-based algorithms vs. Machine learning algorithms (AI)



What is an algorithm?

“An algorithm is a set of rules for getting a specific output from a specific input. Each step must be so precisely defined that it can be translated into computer language and executed by machine”

Source: Knuth, D The Art of Computer Programming (Vol. 1, Fundamental Algorithms, 3rd ed.)

Finiteness

An algorithm must always terminate after a finite number of steps.

Definiteness

Each step must be precisely defined.

Input

An algorithm has zero or more inputs.

Output

An algorithm has one or more outputs, which have a specified relation to the inputs.

Effectiveness

All operations to be performed must be sufficiently basic that they can in principle be done exactly and in finite length of time by someone using pencil and paper.



Donald Knuth



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 2: Algorithmic Complexity



LEAVING CERTIFICATE
COMPUTER SCIENCE





Algorithmic Complexity

Complexity is about analysing algorithms

Choice between algorithms often comes down to efficiency

We need to define objective measures that can be applied to each algorithm

- Execution time?
- Number of statements/instructions executed?
- Number of times a fundamental operation is executed?

Time complexity



Time Complexity

Time complexity gives the number of operations an algorithm performs when processing an input of size n .

An algorithm can have different time complexity values for the same n

We consider 3 cases:

Best Case: minimum number of operations required for a given input

Worst Case: maximum number of operations required for a given input

Average Case: average number of operations required for a given input

Q. Why are computer scientists mostly interested in worst case?

Worst-case analysis lets us make hard guarantees regarding upper bounds on the amount of time it will take a critical process/task to complete.



Big-O

Big O is a notation used in Computer Science to describe the worst-case running time (or space requirements) of an algorithm in terms of the size of its input usually denoted by n .

Big-O notation provides a way to talk about the kind of relationship between the size of the problem and the program running time.

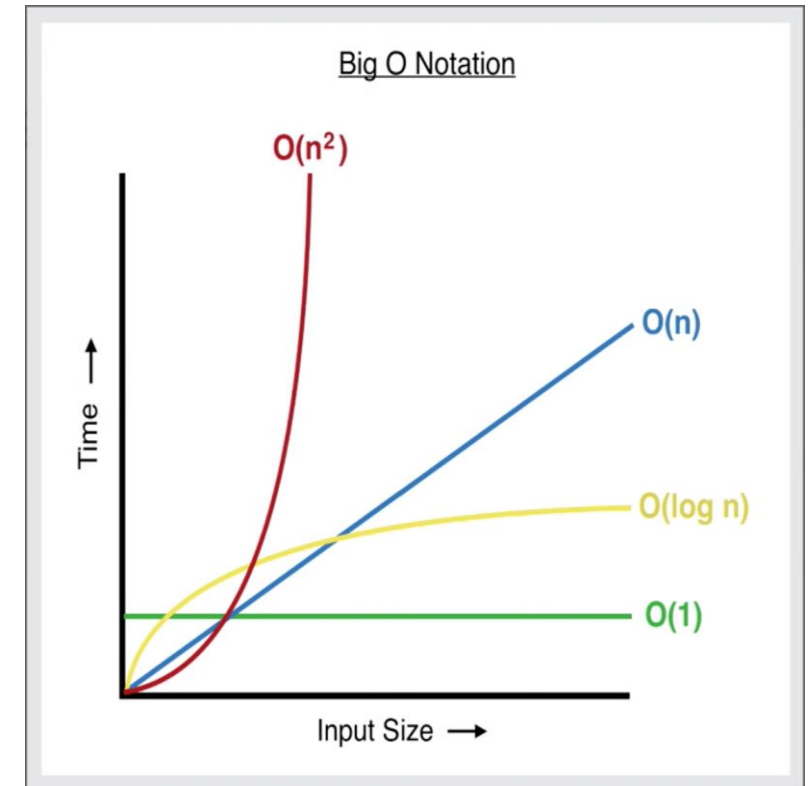
Complexity allows us to classify algorithms (as ‘good’, ‘fair’ or ‘poor’ in terms of performance) and therefore provides a basis to compare algorithms



Big-O

Big-O notation provides a way to talk about the kind of relationship that holds between the size of the problem and the program running time. A shorthand notation for measuring worst-case complexities. It is inexact by design.

$O(1)$	Constant Complexity
$O(n)$	Linear Complexity
$O(n^2)$	Quadratic Complexity
$O(\log_2 n)$	Logarithmic Complexity
$O(n \log_2 n)$	Linearithmic Complexity
$O(2^n)$	Exponential Complexity
$O(n!)$	Factorial complexity
$O(\text{infinity})$	tossing a coin until it lands on heads





Summary: Algorithmic Time Complexity

Always consider the running time and the expected format of the input list before choosing a search or sorting algorithm for a particular problem.

	Best Case	Average Case	Worst Case
Linear Search	$O(1)$	$O(n)$	$O(n)$
Binary Search	$O(1)$	$O(\log_2 n)$	$O(\log_2 n)$
Simple (selection) Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Quicksort	$O(n \log_2 n)$	$O(n \log_2 n)$	$O(n^2)$



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 3: Analysis of Searching and Sorting Algorithms



LEAVING CERTIFICATE
COMPUTER SCIENCE



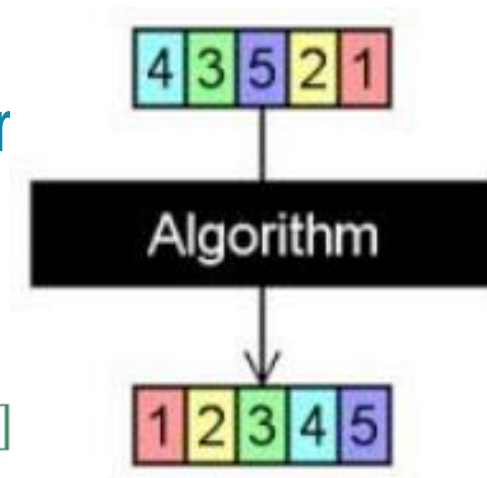


Sorting and Searching

An algorithm that maps the following input/output pair is called a **sorting algorithm**:

Input: A list or array, A , that contains n orderable elements: $A[0, 1, \dots, n - 1]$.

Output: A sorted permutation of A , called B , such that $B[0] \leq B[1] \leq \dots \leq B[n - 1]$



An algorithm that maps the following input/output pair is called a **search algorithm**:

Input: An array, A , that contains n orderable elements: $A[0, 1, \dots, n - 1]$ and some target value commonly referred to as an *argument*.

Output: If the argument is found in A it is conventional to return its zero-based positional offset (i.e. the index) and if the argument is not found some implementations return the length of the list while others return -1 .



Linear Search



```
def linear_search(v, L):  
    for i in range(len(L)):  
        if L[i] == v:  
            return i  
  
    return -1    #not found
```

#Driver code...

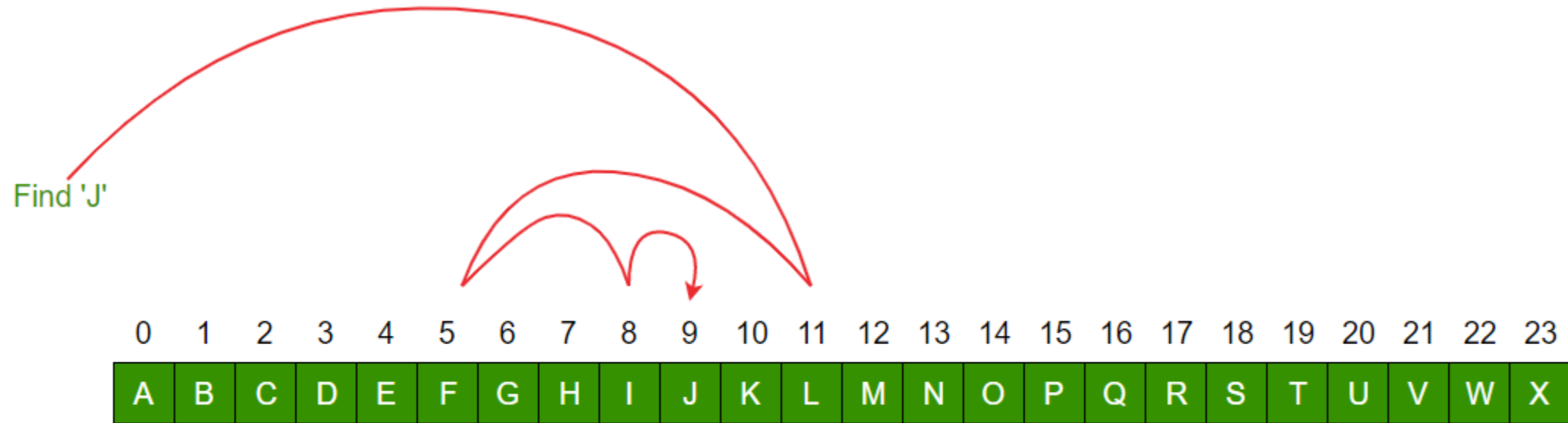
```
keys = [15, 4, 41, 13, 24, 14, 12, 21, 22]  
argument = int(input("Enter a target value:"))  
  
result = linear_search(argument, keys)  
  
if (result != -1):  
    print("%d found at position %d" %(argument, result))  
else:  
    print("%d not found. Return value is %d" %(argument, result))
```

Sample runs ...

```
>>> %Run linear_searches.py  
Enter a target value: 14  
14 found at position 5  
  
>>> %Run linear_searches.py  
Enter a target value: 15  
15 found at position 0  
  
>>> %Run linear_searches.py  
Enter a target value: 12  
12 found at position 6  
  
>>> %Run linear_searches.py  
Enter a target value: 22  
22 not found. Return value is 8
```



Binary Search





Binary Search

```
def binary_search(v, L):  
  
    low = 0  
    high = len(L)-1  
  
    while (low <= high):  
        mid = (low+high)//2  
  
        if L[mid] == v:  
            return mid  
        elif L[mid] < v:  
            low = mid + 1  
        else:  
            high = mid - 1  
  
    return -1
```

```
# Driver code ...  
keys = [2, 4, 5, 7, 8, 9, 12, 14, 17, 19, 22, 25, 27, 28, 33, 37]  
argument = int(input("Enter a target value: "))  
  
result = binary_search(argument, keys)  
  
if (result != -1):  
    print("%d found at position %d" %(argument, result))  
else:  
    print("%d not found. Return value is %d" %(argument, result))
```

Sample runs ...

Sample Run #1
Look for v , 28 in L

Enter a target value: 28
28 found at position 13

Sample Run #2
Look for v , 57 in L

Enter a target value: 57
57 not found. Return value is 16



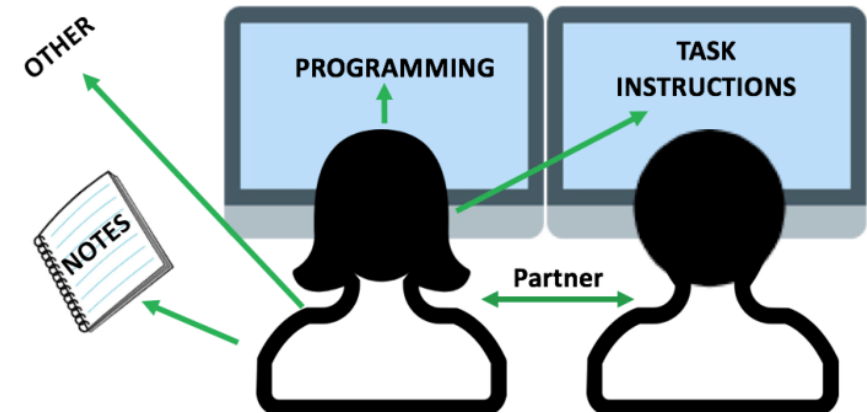
Group Activity: Analysis of Search Algorithms



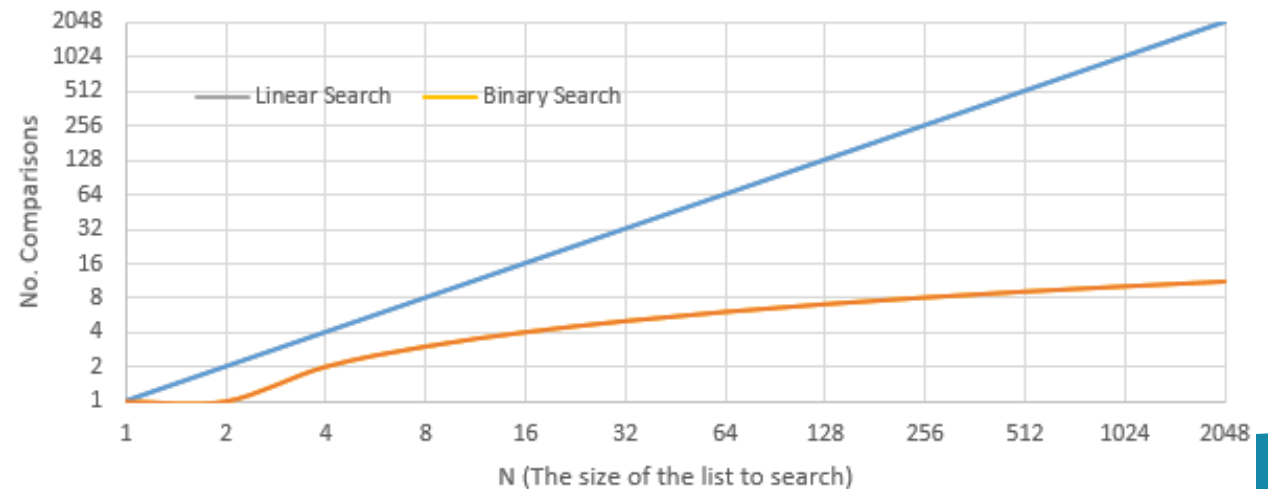


Activity #1: Analysis of Search Algorithms

1. Participants work in pairs (pair programming)
2. Each pair opens the Python programs provided
3. The code is modified according to the instructions in the manual (pages 75-80)



Binary vs. Linear Search



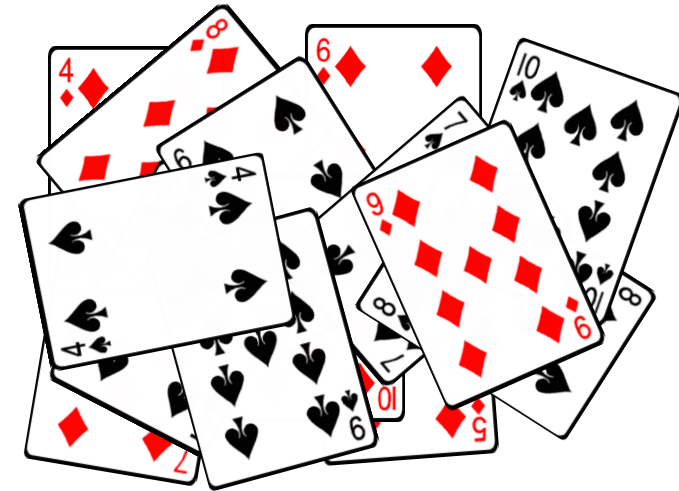
TASK:

Use the analysis framework provided to test the assertion that the binary search is exponentially faster than the linear search (see graph)



Activity #2: Counting Operations

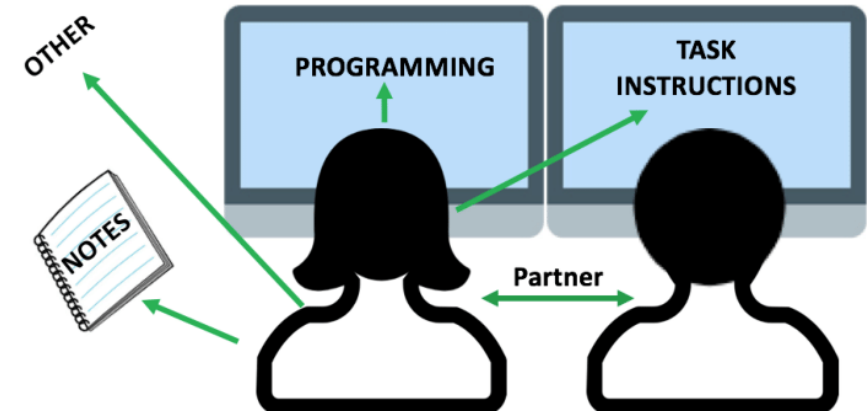
- Spread 8 cards out in a single line on the table – reverse order
- Sort the cards using one of the elementary sorting algorithms counting the total number of comparisons and swaps required
- Repeat for the other two elementary sorting algorithms
- Repeat the process this time starting with the cards already sorted





Activity #3: Analysis of Sort Algorithms

1. Participants work in pairs (pair programming)
2. Each pair opens the Python programs provided
3. The code is modified according to the instructions in the manual
(pages 81-84)

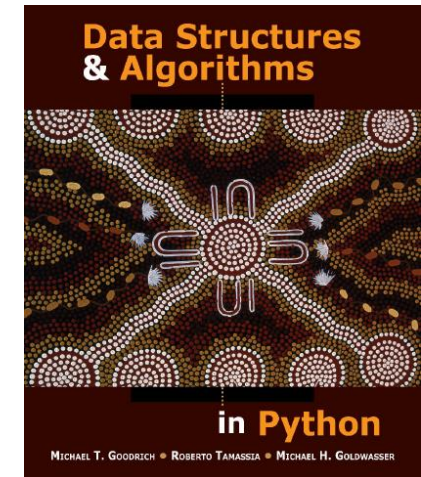
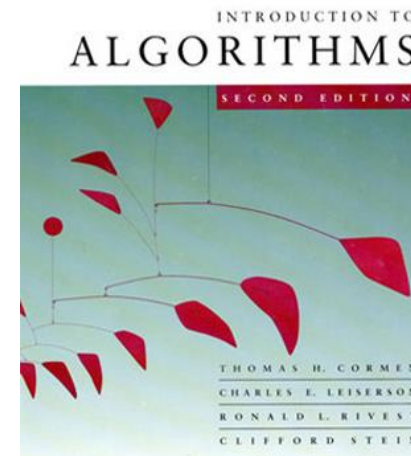
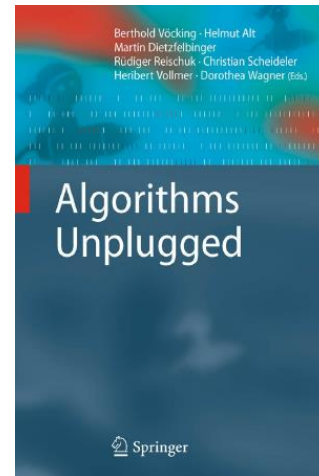


TASKS:

Compare selection sort and insertion sort - best and worst cases
Compare selection sort and bubble sort - best and worst cases



References



<http://courses.cs.vt.edu/~csonline/Algorithms/Lessons/index.html>

<https://www.khanacademy.org/computing/computer-science/algorithms>

<https://algs4.cs.princeton.edu/home>

<https://csunplugged.org/>



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 4: Limits of Algorithms & Heuristic



LEAVING CERTIFICATE
COMPUTER SCIENCE





Algorithmic Complexity: Limits of Algorithms

Example: The Travelling Salesperson Problem (TSP)

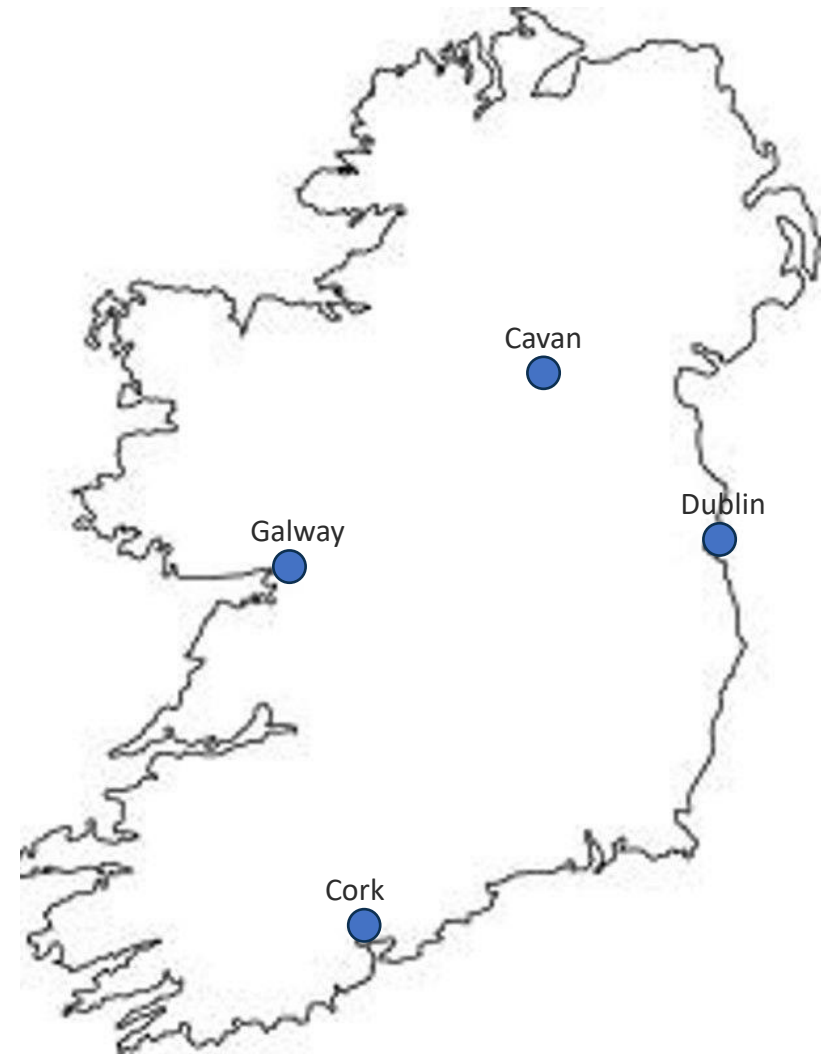
Suppose you decide to ride a bicycle around Ireland you will start in Cavan!

The goal is to visit Dublin, Cork, and Galway before returning to Cavan

	Cavan	Cork	Dublin	Galway
Cavan	-	304	115	168
Cork	304	-	258	206
Dublin	115	258	-	209
Galway	168	206	209	-

What is the best itinerary?

How can you minimise the number of kilometres and yet make sure you visit all the cities?





Real World Applications

For many applications, the number of “cities” (n) can be thousands or more.

While it is not likely anyone would want to plan a bike trip to thousands of cities the solution to finding the shortest tour of a large number of cities can be applied to (is the same as) many important “real world” problems:

- transportation: school bus routes, service calls, delivering meals, post/parcel deliveries, delivery of online purchases (e.g. Amazon)
- manufacturing: an industrial robot that drills holes in printed circuit boards
- design: VLSI (microchip) layout
- communication: planning new telecommunication networks
- space exploration: minimise the use of fuel in targeting and imaging manoeuvres for the pair of satellites involved in NASA Starlight space interferometer program
- biology: to compute DNA sequences

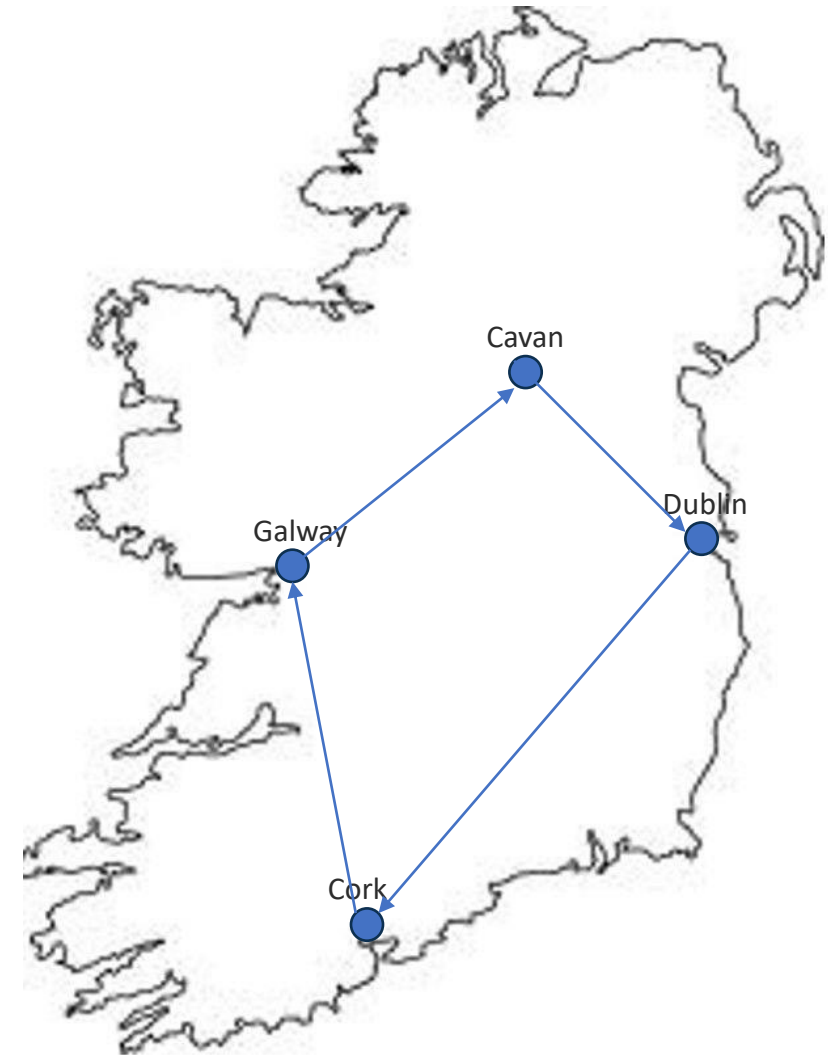




Algorithmic Complexity: Limits of Algorithms

A Brute-Force approach would find all itineraries and then pick the best.

Cavan – Dublin – Cork – Galway – Cavan **747km**





Algorithmic Complexity: Limits of Algorithms

A Brute-Force approach would find all itineraries and then pick the best.

Cavan – Dublin – Cork – Galway – Cavan	747km
Cavan – Dublin – Galway – Cork – Cavan	834km
Cavan – Galway – Cork – Dublin – Cavan	747km
Cavan – Galway – Dublin – Cork – Cavan	939km
Cavan – Cork – Galway – Dublin – Cavan	834km
Cavan – Cork – Dublin – Galway – Cavan	939km



Observations?

(Evaluation and Testing, Computational Thinking – recognising patterns)



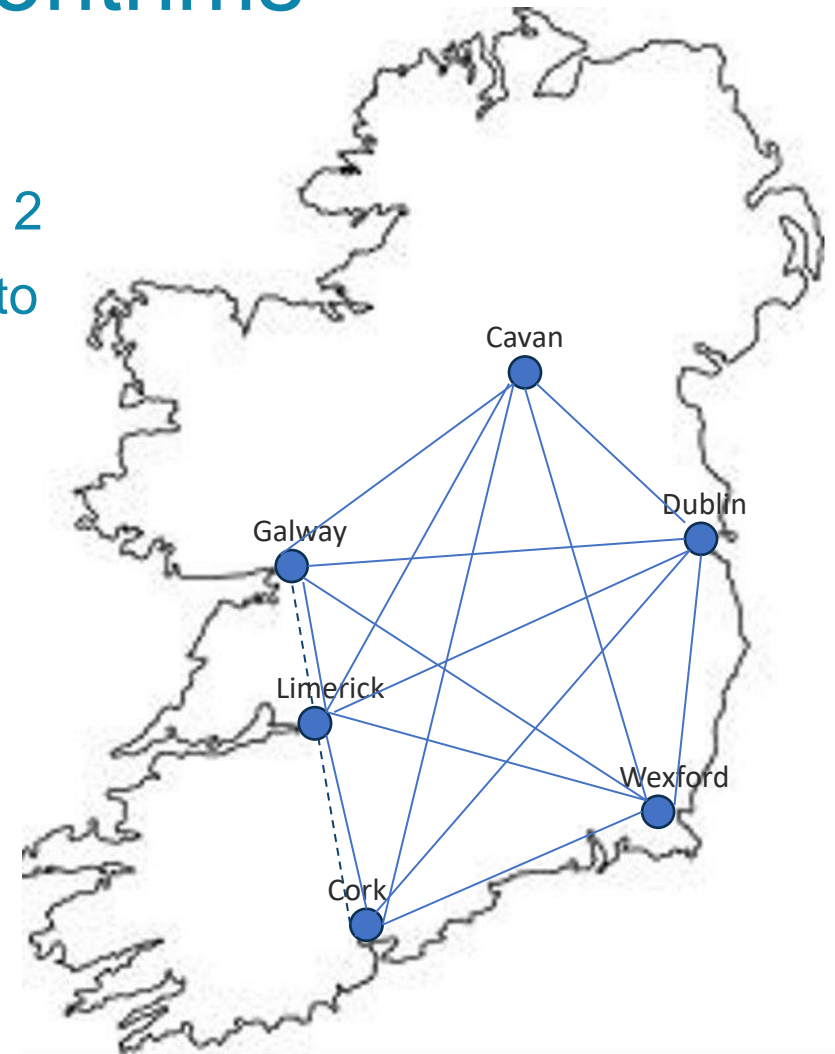
Algorithmic Complexity: Limits of Algorithms

The number of possible tours of a map with n cities is $(n - 1)! / 2$

The number of tours grows incredibly quickly as we add cities to the map

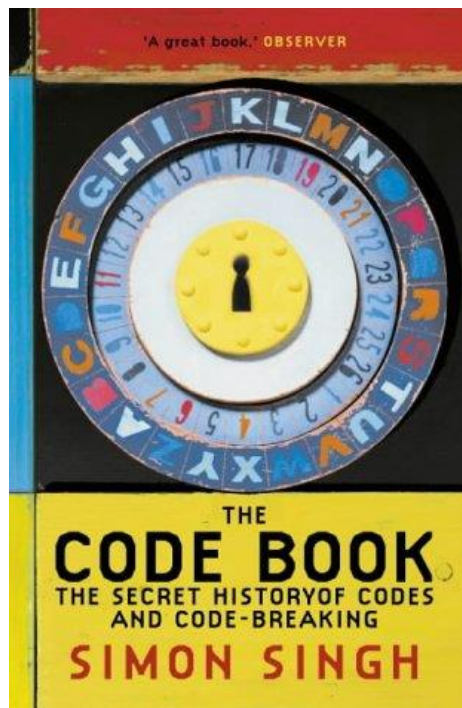
#cities	#tours
5	12
6	60
7	360
8	2,520
9	20,160
10	181,440

The number of tours for 25 cities is 310,224,200,866,619,719,680,000





Algorithmic Complexity: Limits of Algorithms



Pxt dezqlad gxykwedt chwk khxyc kdycdygd

The number of arrangements is
50,000,000,000,000,000,000,000,000,000



Heuristics

An approach to problem solving.

Mental shortcuts that can be used to make a quick decision.

Heuristics are the strategies derived from previous experiences with similar problems.

Not guaranteed to be optimal (but usually take less time than would be required to find an optimal solution). Limitations lie in the trade-offs e.g. correctness vs. performance.

Suitable when finding an optimal solution is impractical or impossible e.g. TSP heuristic may be to pick whatever is currently the best next step regardless of whether that prevents (or even makes impossible) good steps later (known as the greedy algorithm).

Some common examples of heuristics include trial and error, a rule of thumb, an educated guess and intuitive judgement

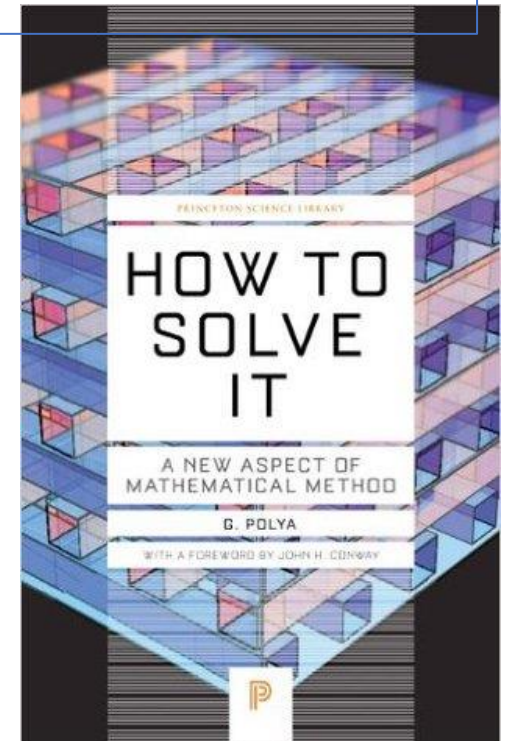


Heuristics

When we cannot solve a problem exactly, one common approach is to use a heuristic instead. A heuristic is a type of algorithm that does not necessarily give a correct answer but tends to work well in practice.

Commonly used heuristics from **George Polya's 1945 book, How To Solve It:**

- If you are having difficulty understanding a problem, try drawing a picture.
- If you can't find a solution, try assuming that you have a solution and seeing what you can derive from that ("working backward").
- If the problem is abstract, try examining a concrete example.
- Try solving a more general problem first (the "inventor's paradox": the more ambitious plan may have more chances of success).





Limitations of heuristics

1. **Optimality:** when several solutions exist for a given problem, does the heuristic guarantee that the best solution will be found?
2. **Completeness:** When several solutions exist for a given problem, can the heuristic find them all?
3. **Accuracy and precision:** Can the heuristic provide a confidence interval for the purported solution?
4. **Execution time:** Is this the best-known heuristic for solving this type of problem?



The Travelling Salesman Problem



The Secret Rules of Modern Living: Algorithms

<https://www.youtube.com/watch?v=kiFfp-HAu64>



We now turn our attention to
focus on a fundamental
question of Computer
Science:

What is computable?





Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Session 5: Computers in Society – Turing Machines



LEAVING CERTIFICATE
COMPUTER SCIENCE





Overview of the session

Part 1	Introduction to Turing machines
Part 2	Group activity: Turing machines
Part 3	Artificial Intelligence: The Turing test



By the end of this session participants will have:

- developed their understanding of **Turing machines** and understand their significance
- participated in an activity to develop a deeper understanding of how Turing machines operate
- participated in an activity on **artificial intelligence**



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 1: Introduction to Turing Machines



LEAVING CERTIFICATE
COMPUTER SCIENCE



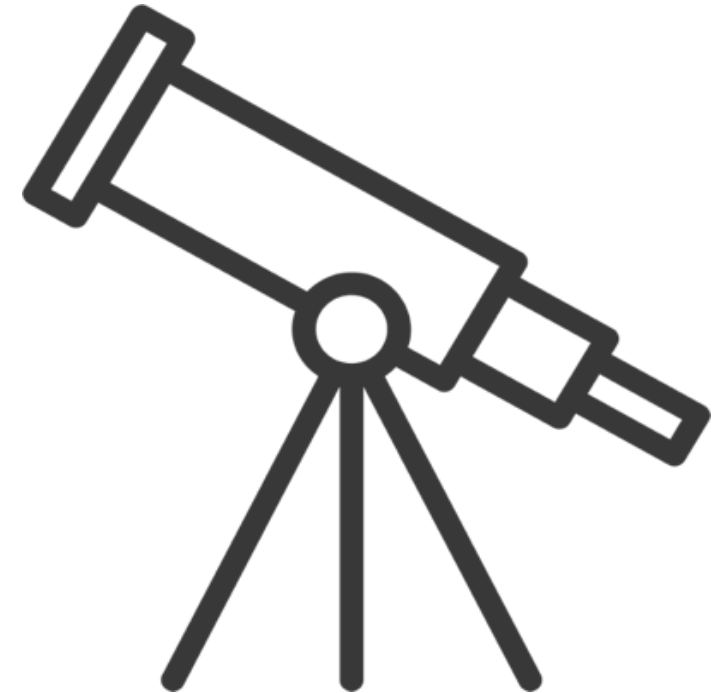
A fundamental question of Computer Science...



Oide

What is computable?

How do we define computability?





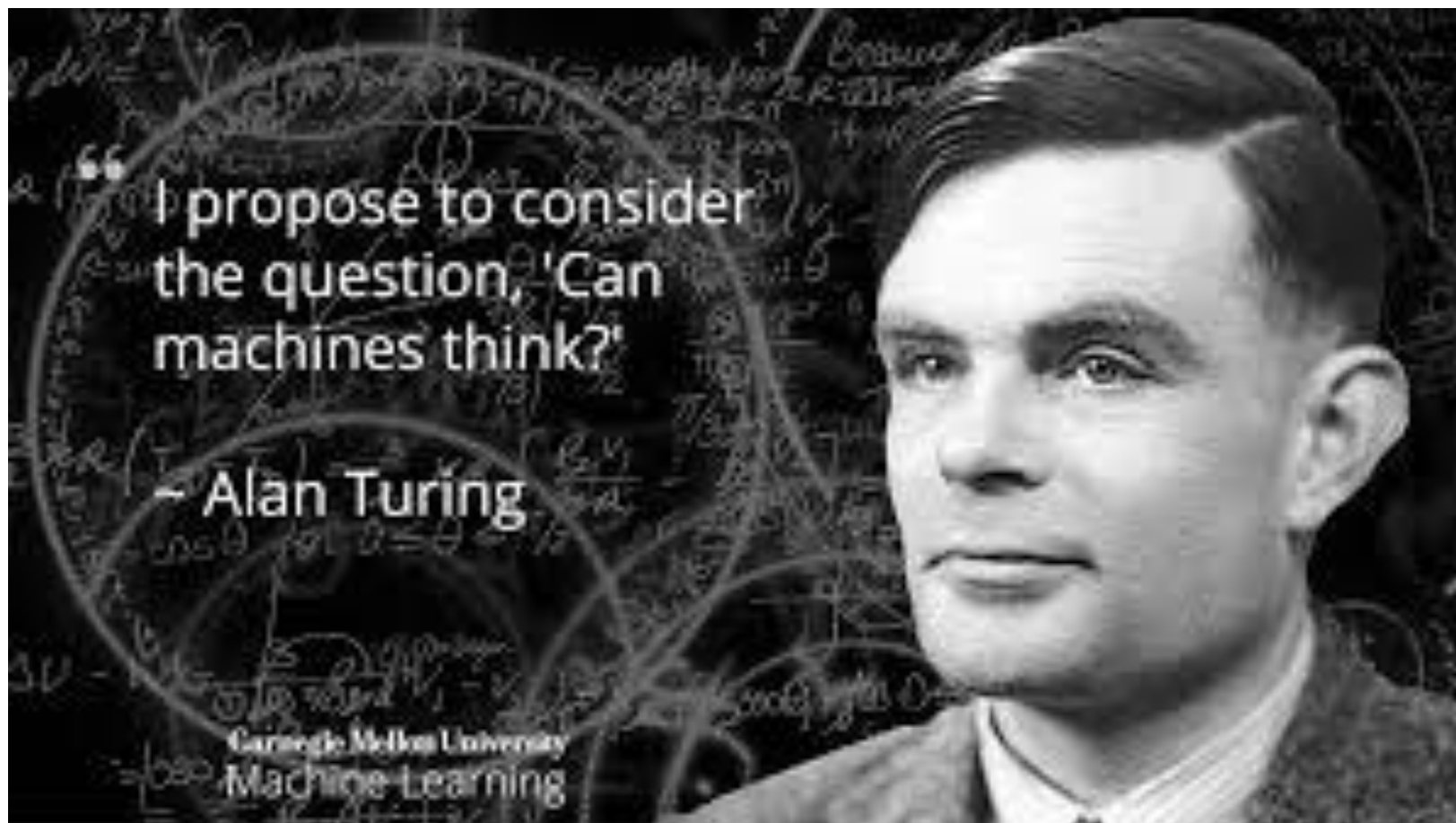
Turing Machines

Computability

Code breaking

Computer design

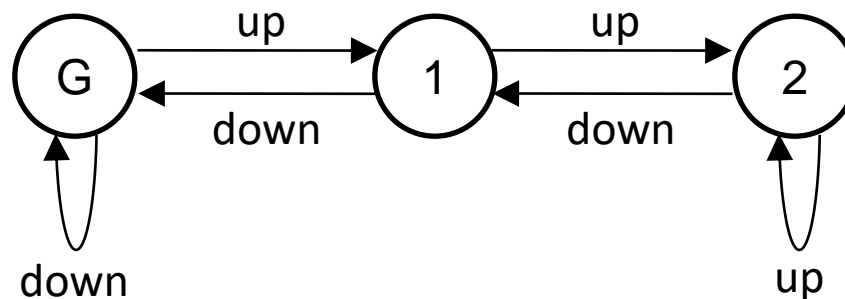
Artificial Intelligence





Introducing Turing machines

The illustration below is of an elevator represented as a finite-state machine



Circles represent states (in this case floors).

Arrows between circles represent transitions between states.

The labels on each transition represents the button press event.

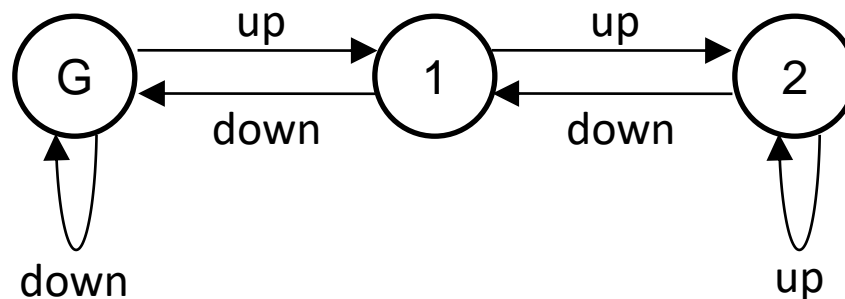
What happens when we are on the ground floor and press the UP button?

What happens when we are on the ground floor and press the DOWN button?



Introducing Turing machines

The illustration below is of an elevator represented as a finite-state machine



Assuming we start in state 1, what would the following input sequence yield?

UU DDDD UU D

The notion of a Turing machine is not too unlike this...

Given a Finite State Machine and an input, we can determine an output



Introducing Turing machines



<https://www.youtube.com/watch?v=dNRDvLACg5Q&t=2s>



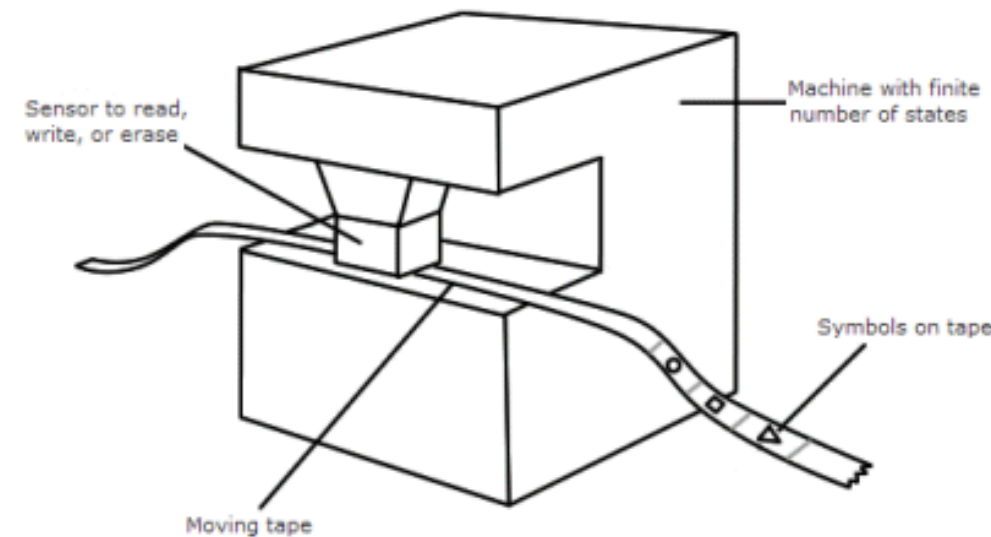
Introducing Turing Machines

The Turing Machine (TM) was invented in 1936 by Alan Turing.

It is a basic abstract symbol-manipulating device that can be used to simulate the logic of any computer that could possibly be constructed.

Although it was not actually constructed by Turing, its theory yielded many insights.

Anything that is possible to (mathematically) compute could be programmed on a Turing machine.

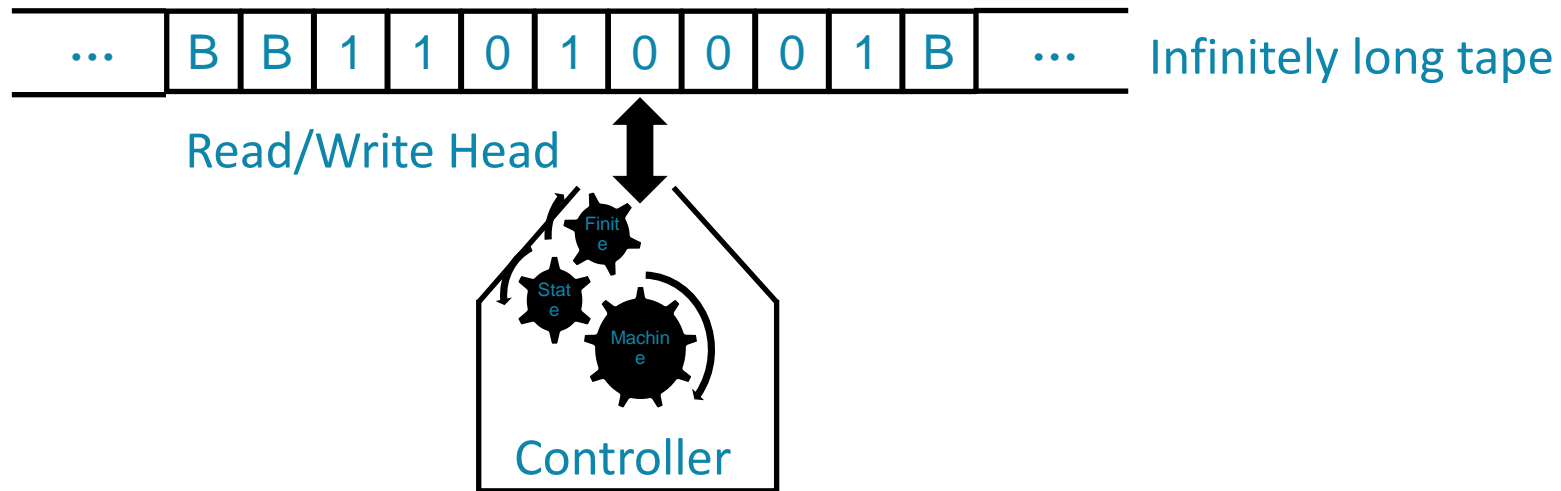




Turing Machines - Introduction

A Turing Machine consist of three components as follows:

1. An infinitely long tape made up of individual cells. Each cell can contain a single character – typically 1, 0, or B (blank)
2. A read/write head pointed at an individual cell
3. A controller (aka finite-state machine) which instructs the read/write head what to do



A schematic representation of a Turing Machine



Turing Machines - Operation

Initially the tape is inscribed with a sequence of characters – called the input

For example:



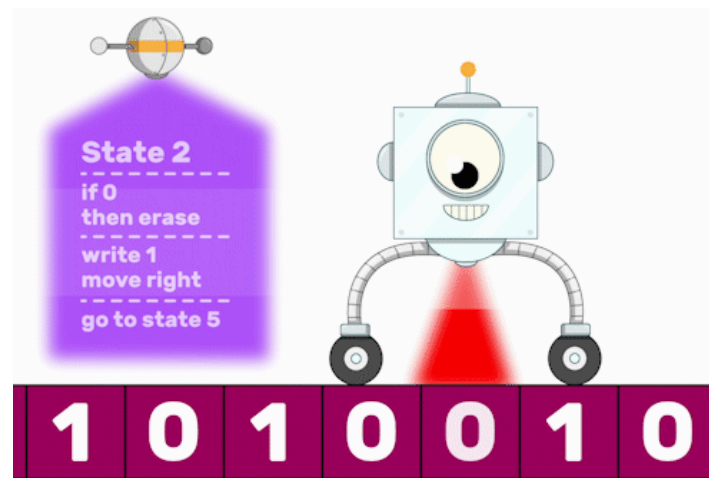
The operation of the Turing Machine is controlled by the finite-state machine (controller).

The operation takes place as a sequence of steps known as transitions.

The controller decides for a given (input character, state) pair, the (output character, state) pair - known as a transition.

Each transition involves:

- Reading
- Writing
- Moving
- Updating (state)



Turing Machines - Operation



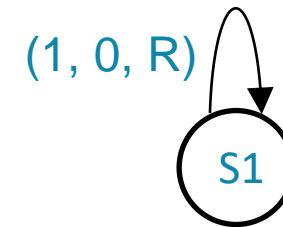
Transitions can be expressed using:

state transition tables

OR

state transition diagrams

Current State	Read (input)	Write (output)	Direction to move r/w head	Next State
S1	1	0	Right	S1



The above state transition table and diagram shows a single transition which says:

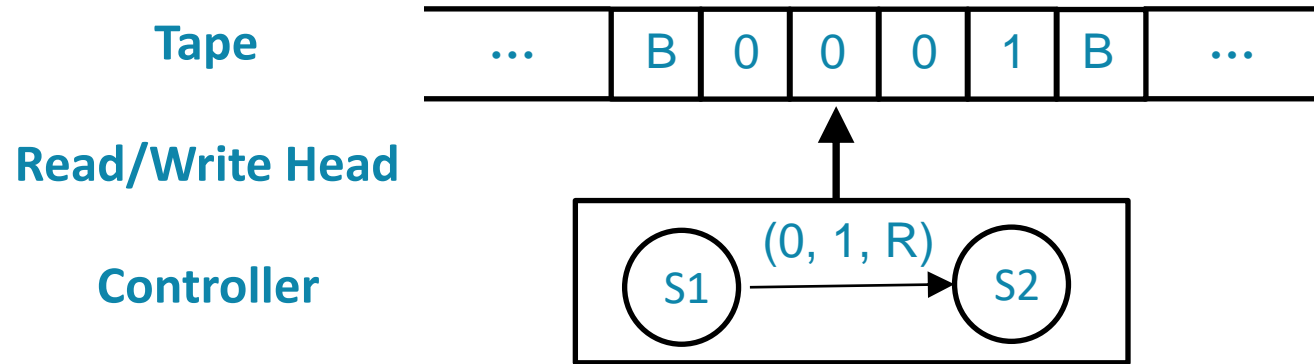
“When in state S1 and the symbol being read is a one, write a zero, move right and remain in state S1”.

if (state == S1) and (character == 1):
write 0
move right
set state to S1

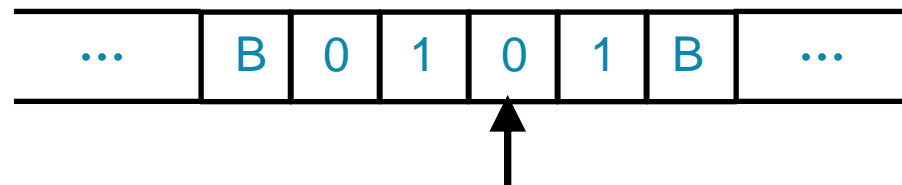
Turing Machines - Operation



The illustration below depicts a TM which defines a transition from state S1 to S2 when the current symbol being read in a zero.



After the transition has been completed, the symbol zero has been replaced with a 1, the read/write head has been moved right and the new state is set to S2.



The result of the computation (output) is the sequence of characters left on the tape if and when the Turing Machine halts.



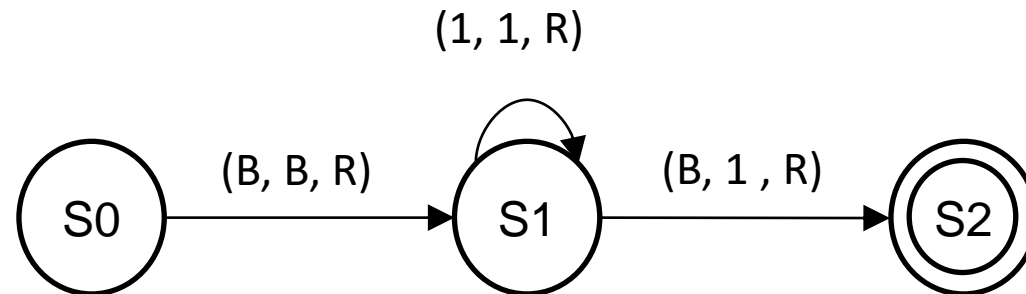
Turing Machines – States

At any given time, a Turing Machine is said to be in a particular state. States are usually denoted by the letter S followed by a number e.g. S2 is taken to mean state two.

S0 is conventionally used to denote the initial state. This is the state the Turing Machine is in before it starts to operate.

A double circle is used to denote the final or *halting state*. This is the state the Turing Machine is in when it finishes.

For example:





A fundamental question of Computer Science...

What is computable?

How do we define computability?

Answer: A task is computable if it can be carried out by a Turing Machine.





Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 2: Group activity: Turing machines



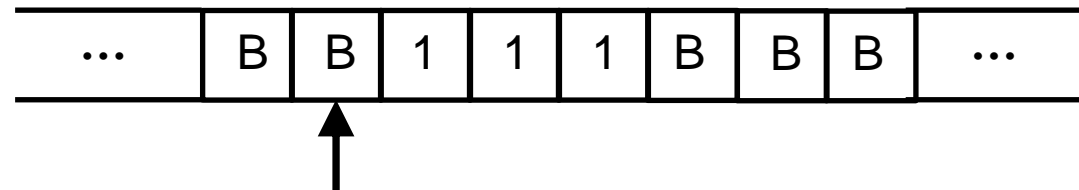
LEAVING CERTIFICATE
COMPUTER SCIENCE



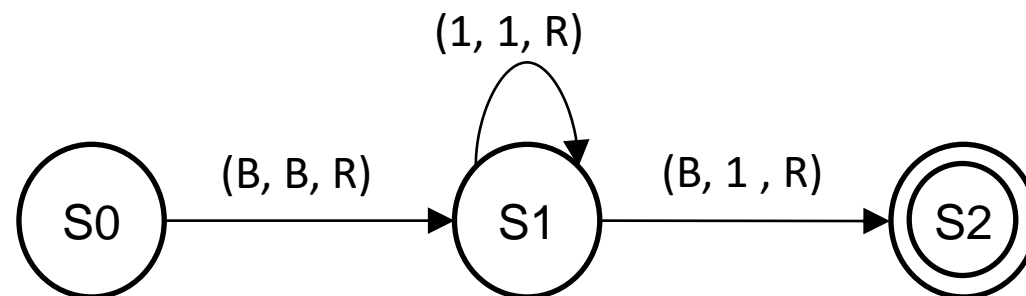
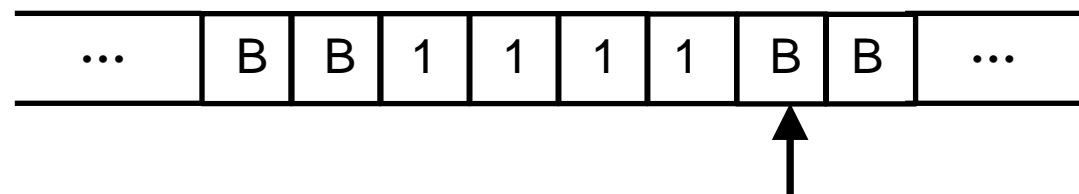


Turing Machines – Example (unary increment)

Test input: 111 (3)



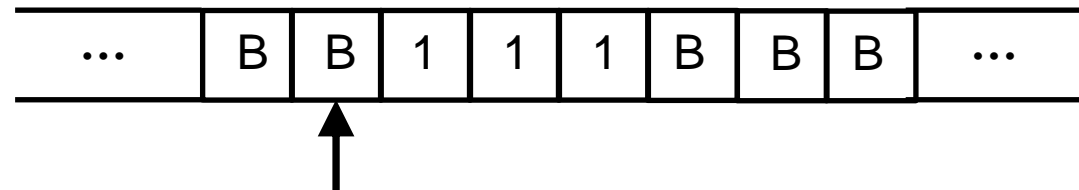
Required output: 1111 (4)



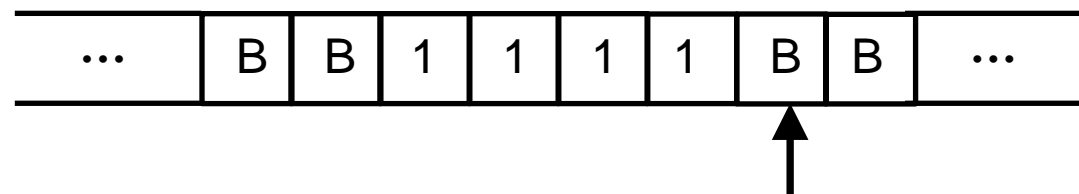


Turing Machines – Example (unary increment)

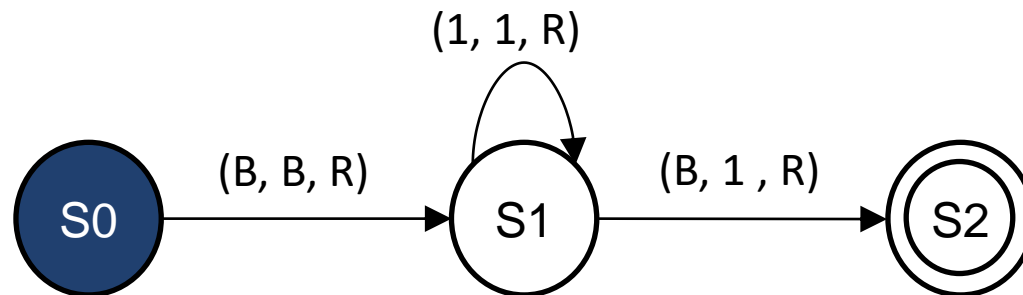
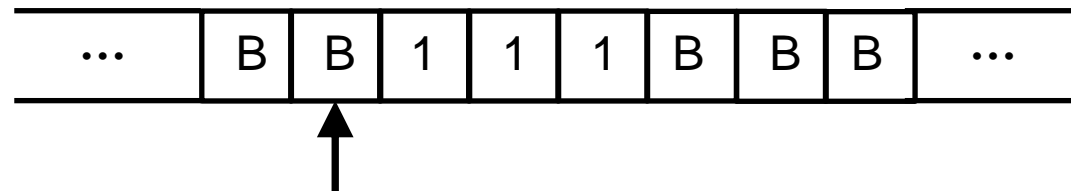
Test input: 111 (3)



Required output: 1111 (4)



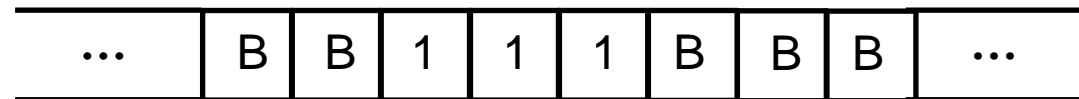
State: S0



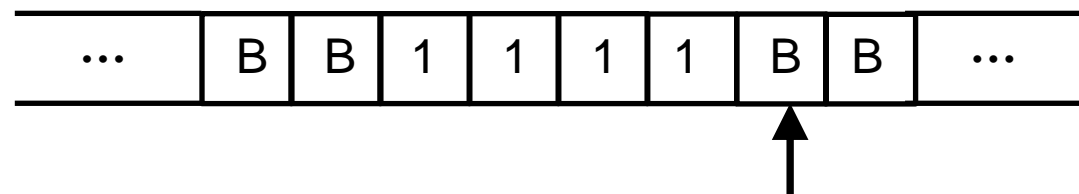


Turing Machines – Example (unary increment)

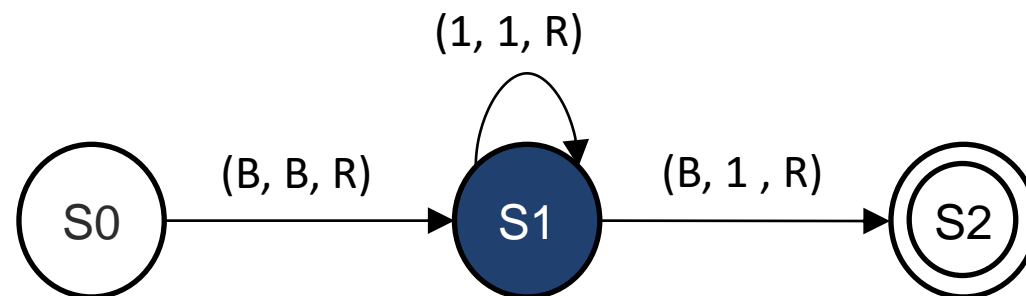
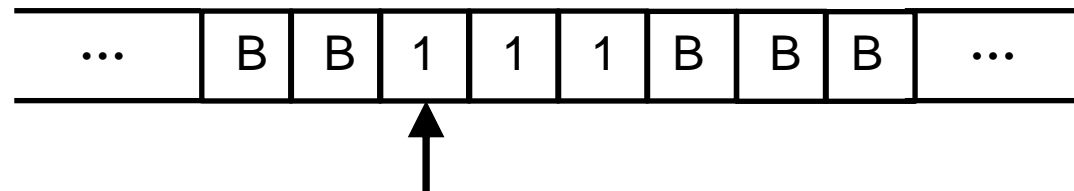
Test input: 111 (3)



Required output: 1111 (4)



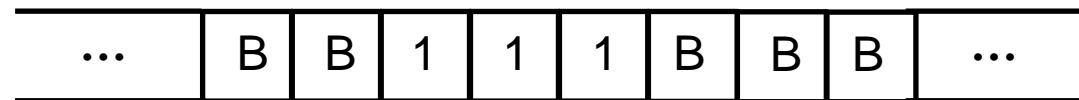
State: S1



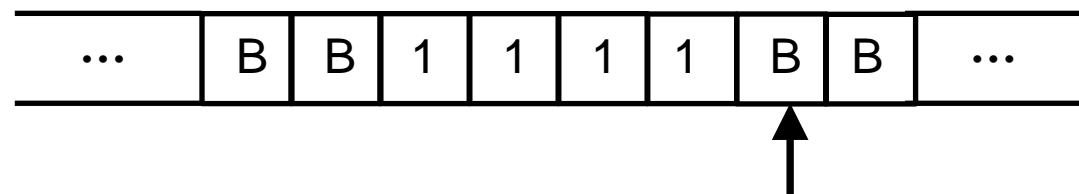


Turing Machines – Example (unary increment)

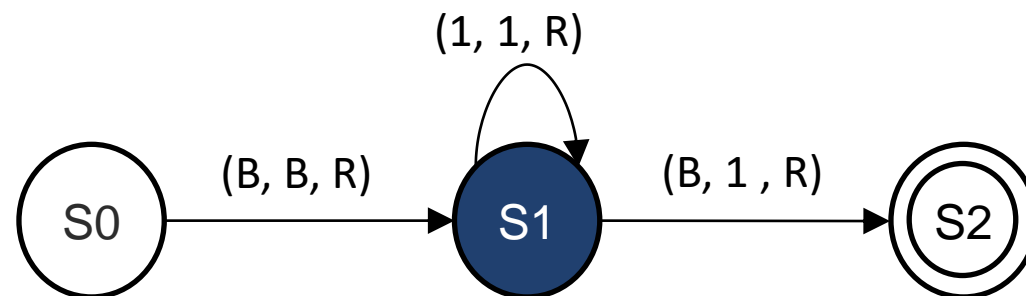
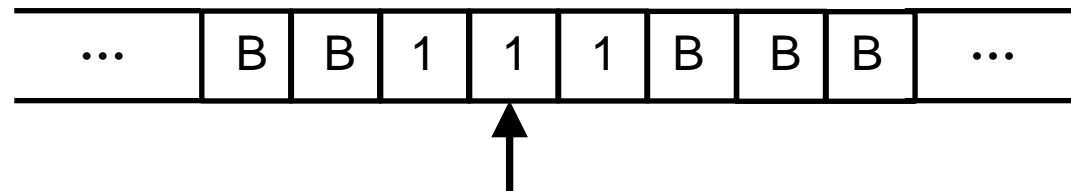
Test input: 111 (3)



Required output: 1111 (4)



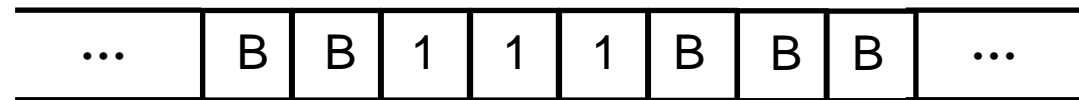
State: S1



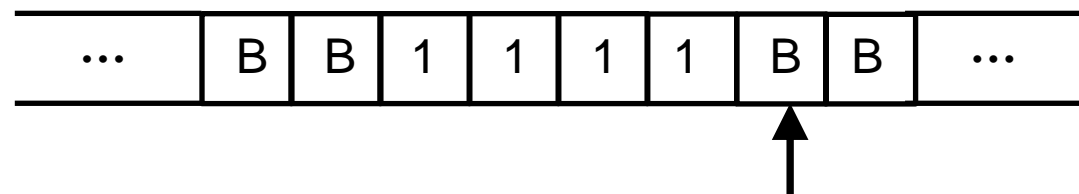


Turing Machines – Example (unary increment)

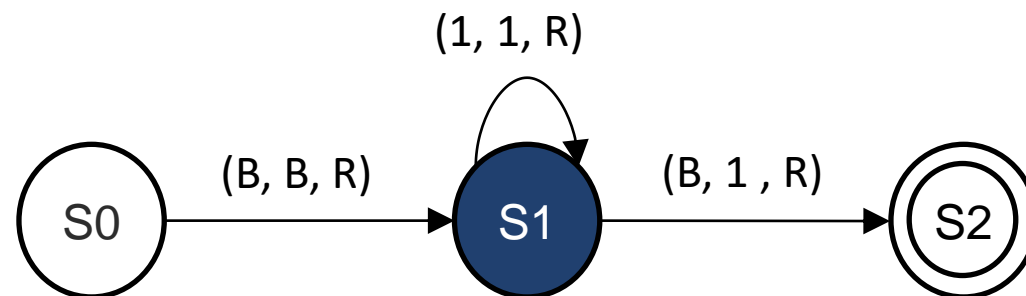
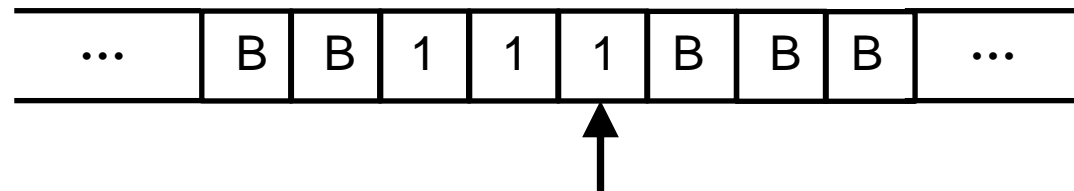
Test input: 111 (3)



Required output: 1111 (4)



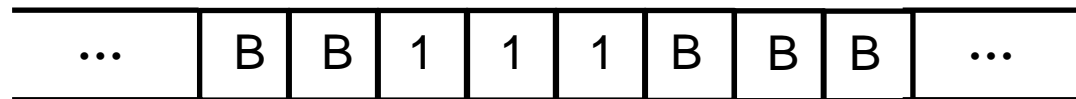
State: S1



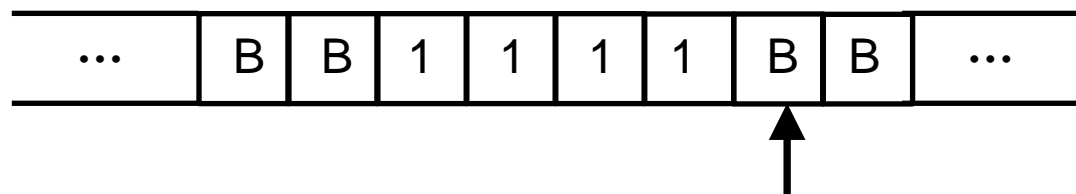


Turing Machines – Example (unary increment)

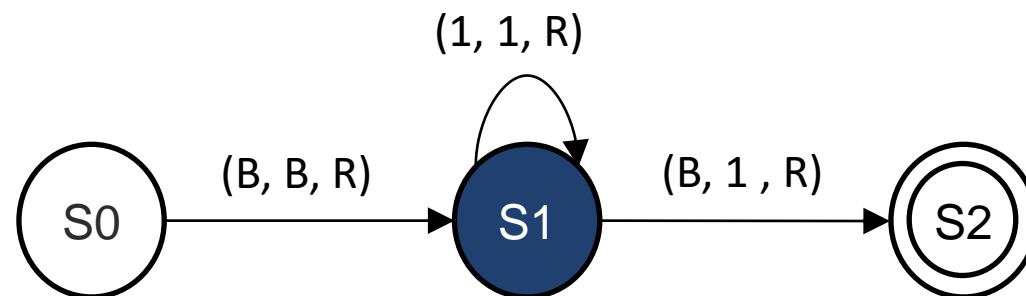
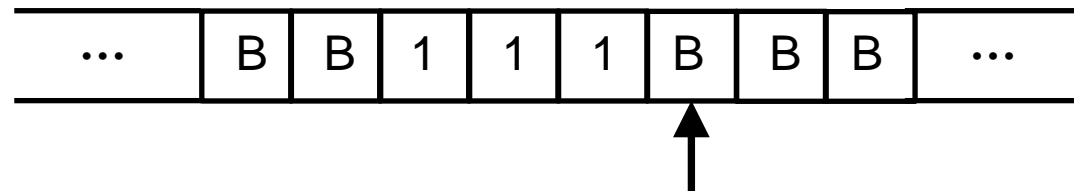
Test input: 111 (3)



Required output: 1111 (4)



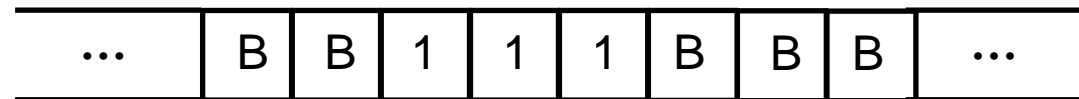
State: S1



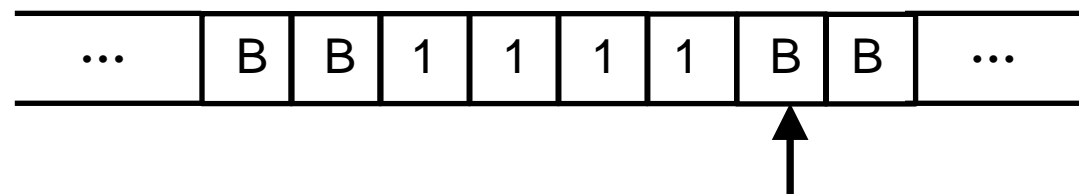


Turing Machines – Example (unary increment)

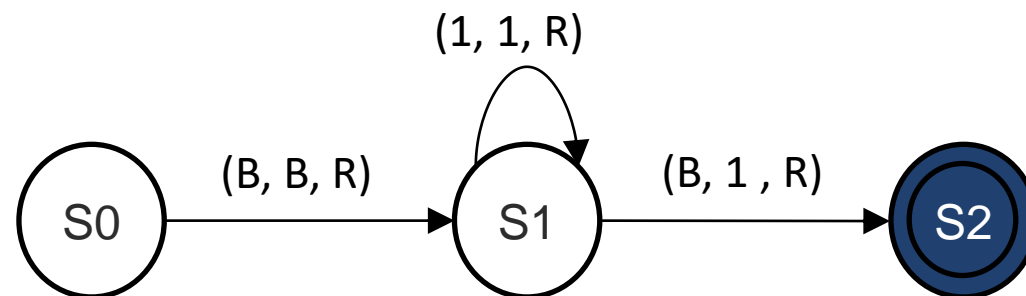
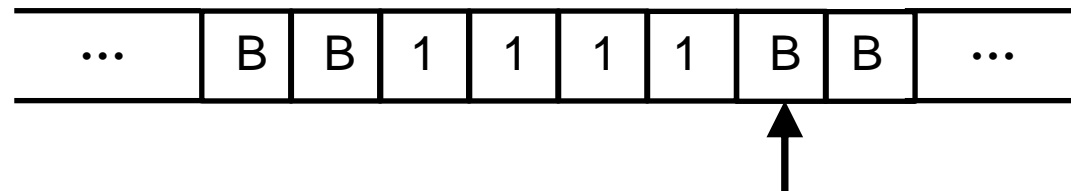
Test input: 111 (3)



Required output: 1111 (4)



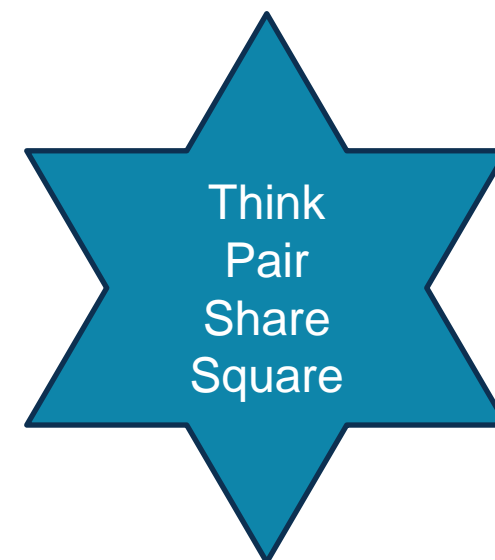
State: S2





Turing Machine Activity

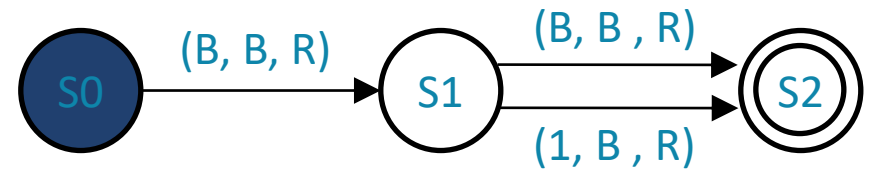
Each group will trace through the operation of the Turing Machines assigned.



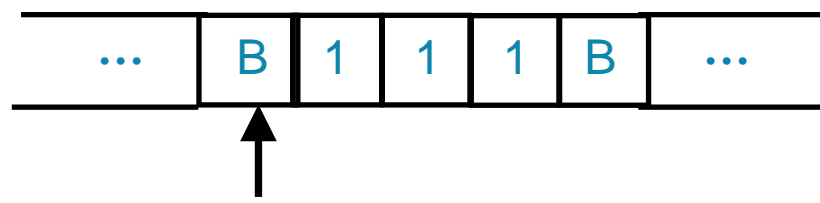


Turing Machines – Activity – Problem #1

Initial State: S_0



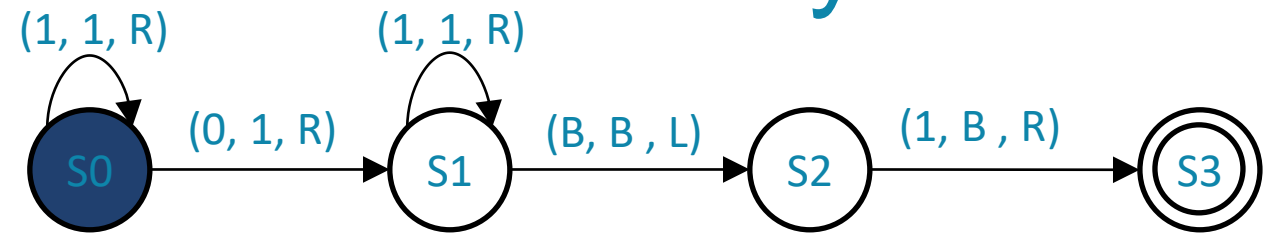
Test input: **B111B**



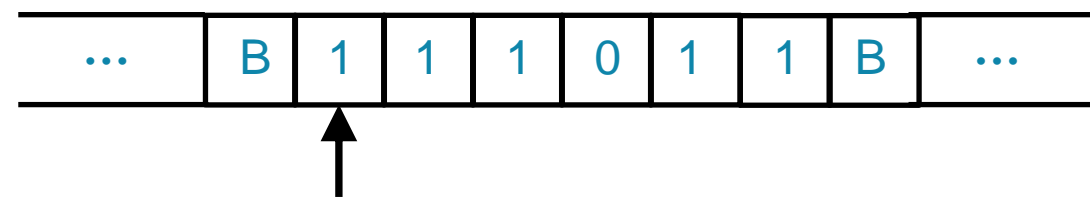


Turing Machines – Activity – Problem #2

Initial State: S0



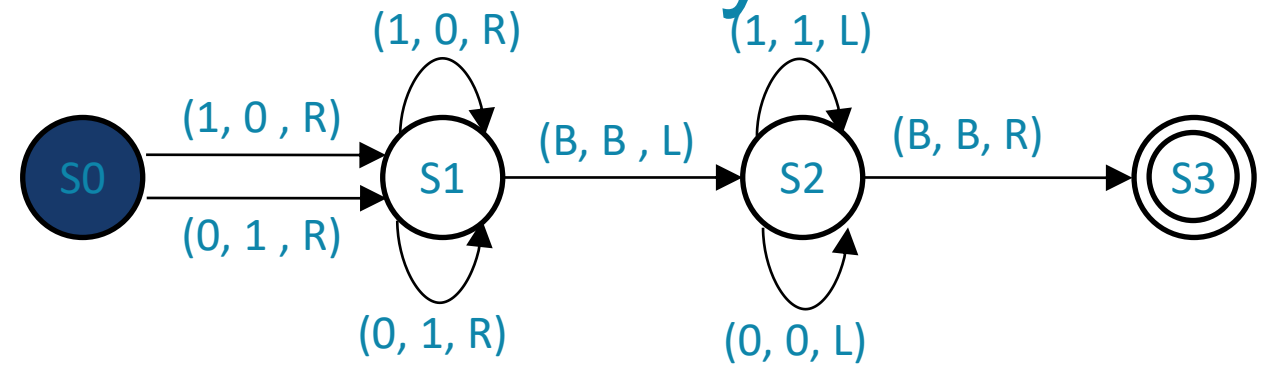
Test input: 111011



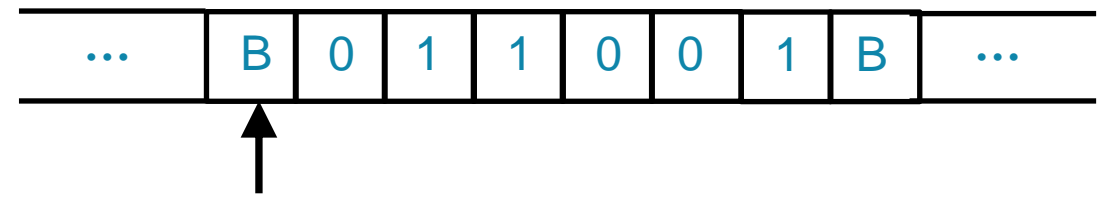


Turing Machines – Activity – Problem #3

Initial State: S0



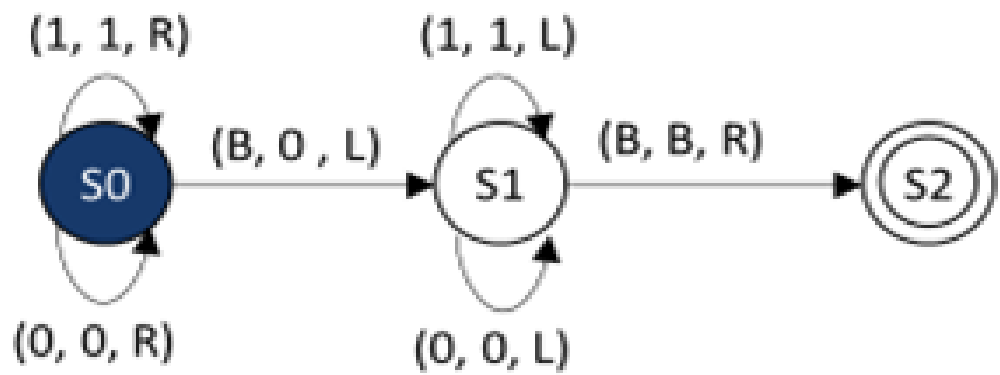
Test input: B011001B



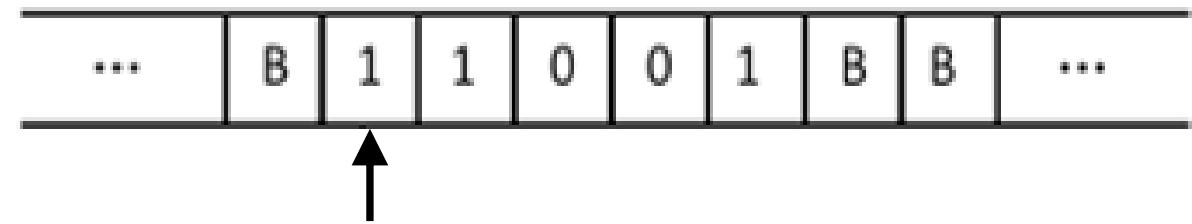


Turing Machines – Activity – Problem #4

Initial State: S0



Test input: B11001BB





Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 3: Artificial Intelligence



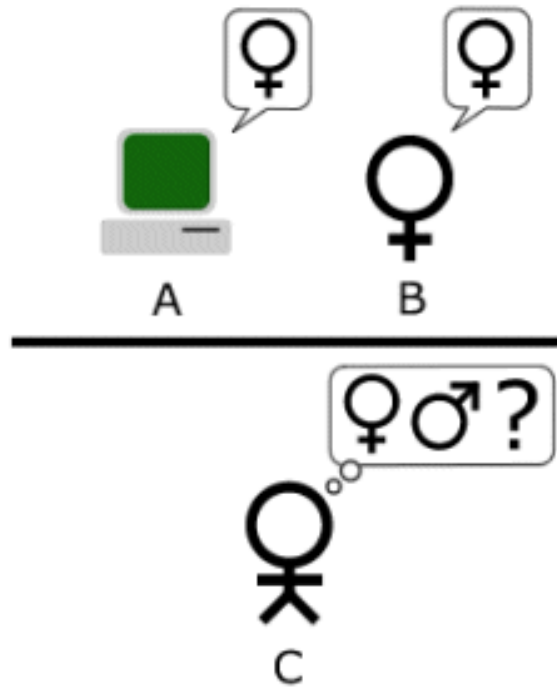
LEAVING CERTIFICATE
COMPUTER SCIENCE





The Turing Test

The Turing test of artificial intelligence proposes a simple game where a hidden computer A and a person B converse with another person C. If C is unable to distinguish which he is conversing with, then the computer can be said to be able to "think".





A Test



<https://www.youtube.com/watch?v=ijwHj2HaOT0&t=244s>



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Session 6: Curriculum Planning



LEAVING CERTIFICATE
COMPUTER SCIENCE





Overview of the session

Part 1	Introduction to Curriculum Planning
Part 2	Group activity: experiencing LOs through the lens of the ALTs
Part 3	Presenting a Curriculum Planning tool
Part 4	Wrap up and conclusions

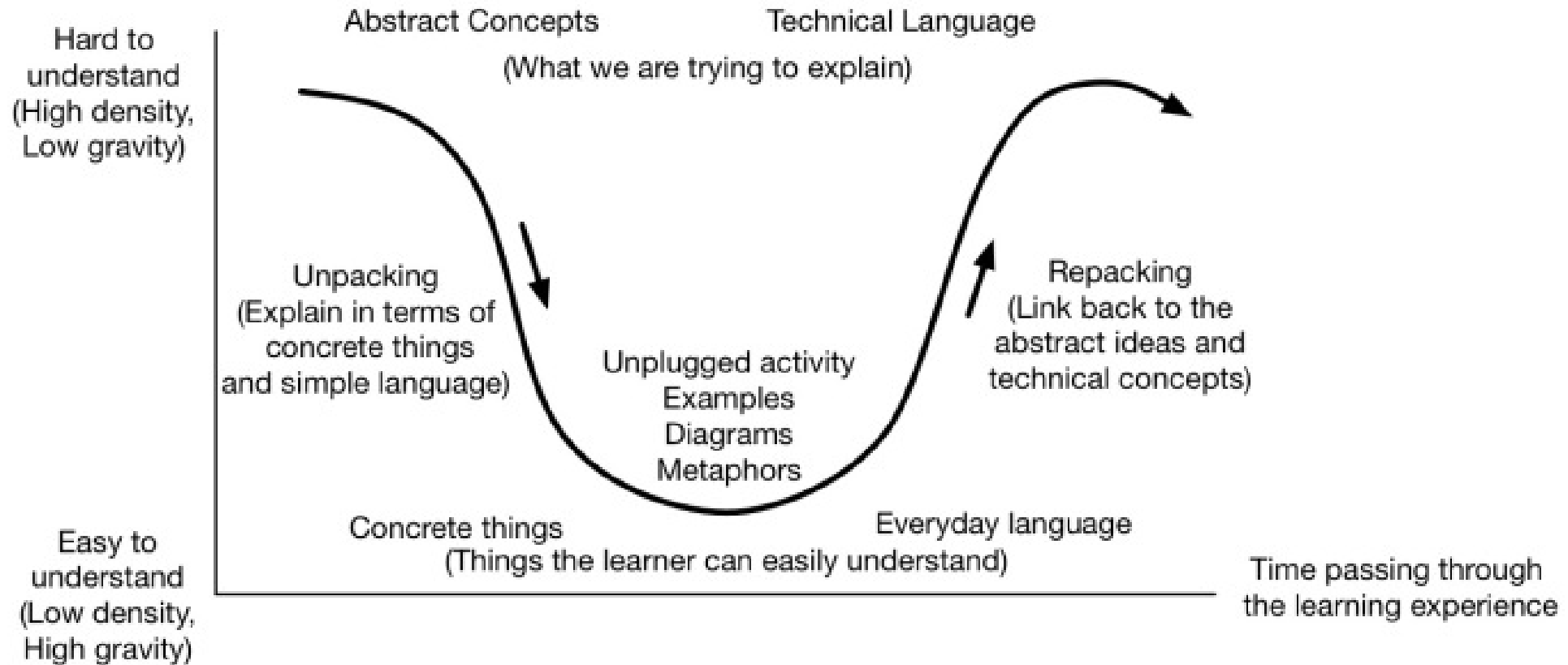


By the end of this session participants will have:

- reflected on the LOs that have been experienced through the ALTs up to now
- collaborated on **Bubbl.us** to develop a concept map of the **LOs that have been and could be experienced** through a particular ALT
- **given and received feedback** on the potential LOs that could be experienced through a particular ALT
- engaged with and used **bespoke spreadsheet technology** to enhance ALT planning practice alongside concept mapping ideas



Semantic Waves



<https://teachinglondoncomputing.org/semantic-waves/semantic-waves-tip9/>



Bubble Sort Dance



<https://youtu.be/lv3vgjM8Pv4>



Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 1: Introduction to Curriculum Planning



LEAVING CERTIFICATE
COMPUTER SCIENCE

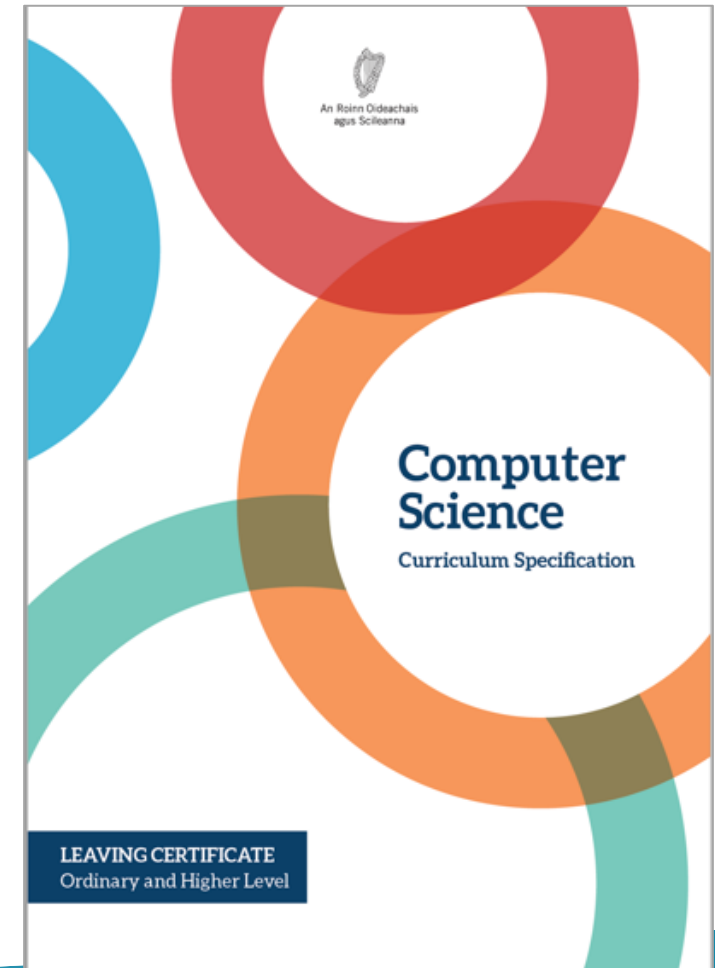
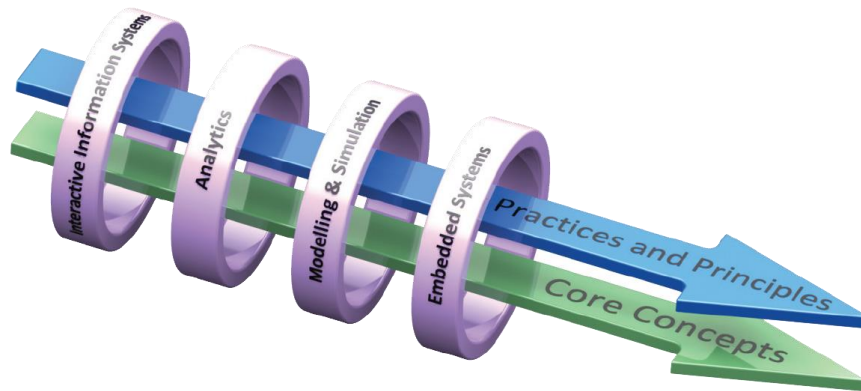




Leaving Certificate Computer Science

The strand 3 applied learning tasks that students undertake collaboratively during the two years of the course, provide significant engaging opportunities for students to work within the practices and principles of computer science and to apply the core concepts in authentic situations.

(Computer Science Curriculum Specification Pg. 15)



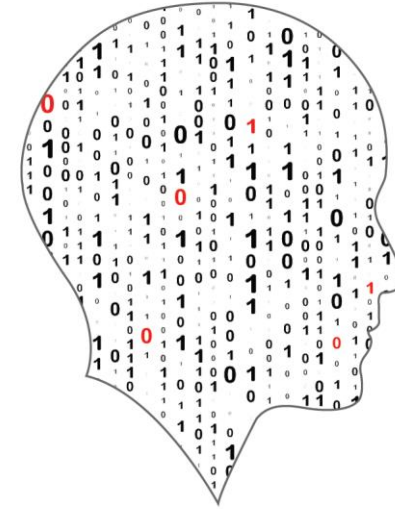


Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 2: Experiencing LOs through the lens of the ALTs



LEAVING CERTIFICATE
COMPUTER SCIENCE





Considering curriculum planning

What **learning outcomes** are we hoping our students will experience – or build towards – in this ALT?

What **learning experiences** can we offer to our students to achieve this?

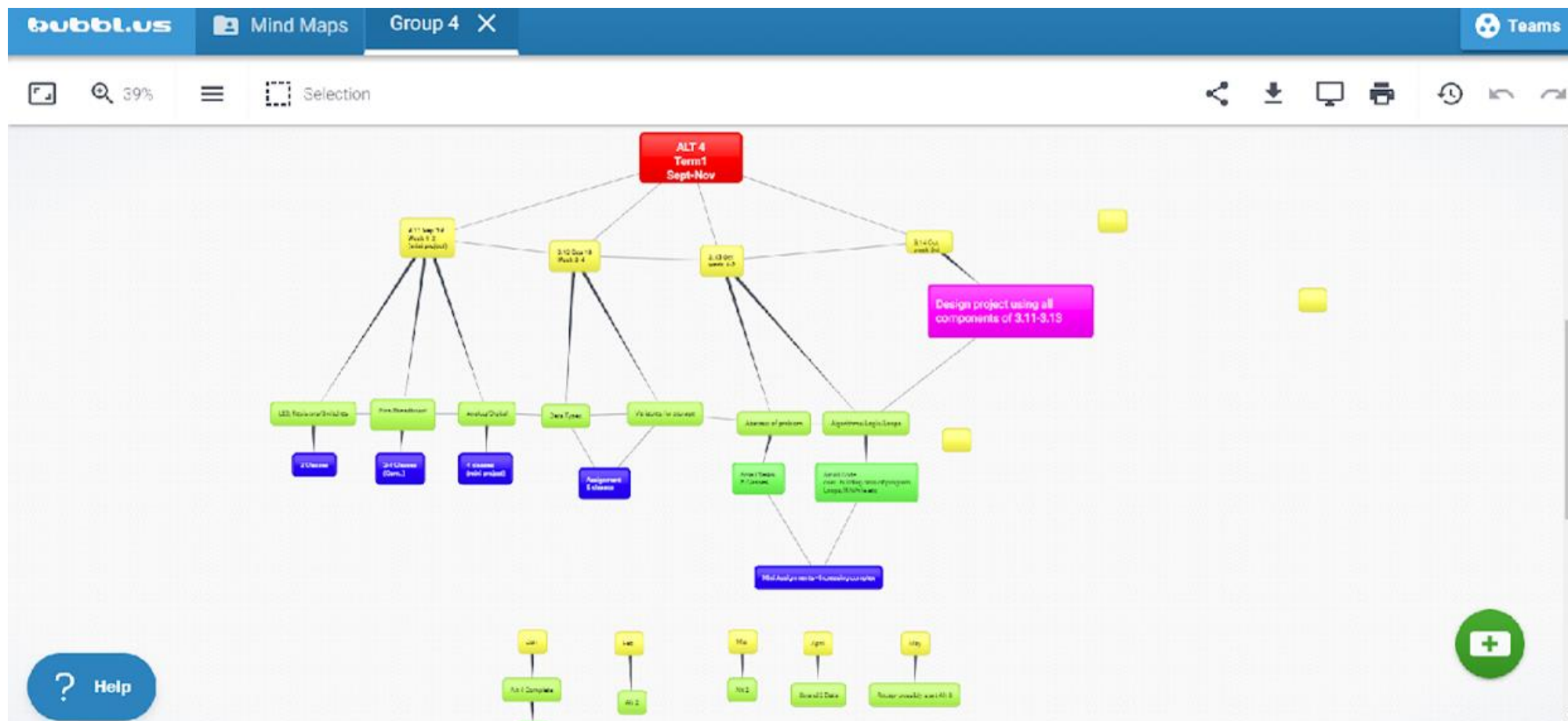
What stimulus can we provide to **enhance** the learning?

What unique considerations should we make for our particular context (class schedule, access to technology, individual student need and, specifically, considerations of **Special Education Needs**)?





bubbl.us





Group activity: Developing a mind map for an ALT

Click on the link to the **Bubbl.us** template for your particular ALT

Nominate a **spokesperson**

Reflect and discuss the LOs your students have experienced (and could experience in the future) through the lens of this ALT

Add these LOs to the mind map at the appropriate node

Add **learning experiences** to your mind map that incorporate these LOs

Prepare **feedback** for the main group





Group Activity Feedback

Click on the link to the **Bubbl.us** template for your particular ALT

Nominate a **spokesperson**

Reflect and discuss the LOs your students have experienced (and could experience in the future) through the lens of this ALT

Add these LOs to the mind map at the appropriate node

Add **learning experiences** to your mind map that incorporate these LOs

Prepare **feedback** for the main group





Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 3: Introduction to a curriculum planning tool

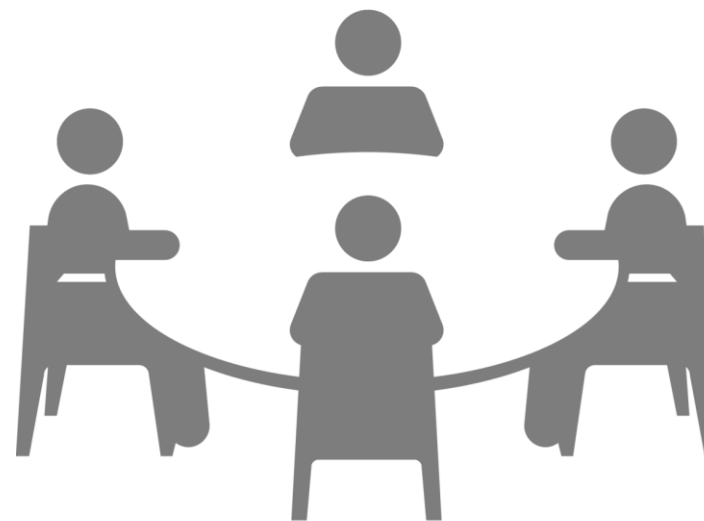
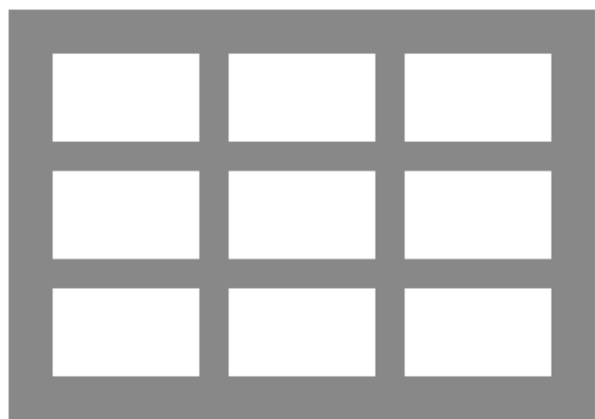


LEAVING CERTIFICATE
COMPUTER SCIENCE



Using the curriculum planning tool

- <https://tinyurl.com/LCCSplanning>



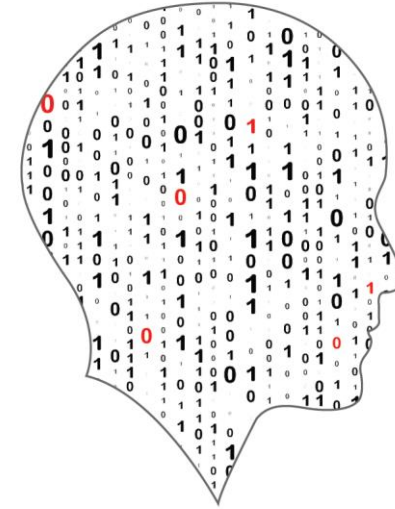


Oide

Tacú leis an bhFoghlaim
Ghairmiúil i measc Ceannairí
Scoile agus Múinteoirí

Supporting the Professional
Learning of School Leaders
and Teachers

Part 4: Wrap-up and conclusions



LEAVING CERTIFICATE
COMPUTER SCIENCE





Conclusions

LCCS is difficult (for students to learn and teachers to teach)

Pedagogies are proven to work

Planning learning around ALTs is key

Constructivist approach is important

Growth mindset is *at least* as important as natural ability

Student-centric approach (guide-on-the-side rather than a sage-on-the-stage approach)



“The teacher should help, but not too much and not too little, so that the student shall have a reasonable share of the work” and, “If the student is not able to do much, the teacher should leave him at least with some illusion of independent work.”

George Polya, “How To Solve It”