

# Communicating Computer Science

# 1

## National Curriculum in Computing

*7 October 2022*

Sebastian Wild

# 1 National Curriculum in Computing

- 1.1 Educational System in England & Wales
- 1.2 The Role of Computer Science
- 1.3 Computing in KS 3
- 1.4 Widening Participation

## **1.1 Educational System in England & Wales**

# Progression in State-Funded Schools

Year	Age	Key Stage	School form	Exams
Reception / Foundation 2	4–5	KS0		
Year 1	5–6	KS1	Primary	
Year 2	6–7			
Year 3	7–8	KS2		
Year 4	8–9			
Year 5	9–10			
Year 6	10–11			SATS, 11-plus
Year 7	11–12	KS3	Secondary	
Year 8	12–13			
Year 9	13–14			
Year 10	14–15	KS4		
Year 11	15–16			GCSE
Year 12	16–17	KS5	Sixth form	
Year 13	17–18			A-Levels

# GCSEs

## General Certificate of Secondary Education

▶ exams at end of Year 11

▶ Grades

▶ 9–5 “strong pass”  
before 2019: A\*–C

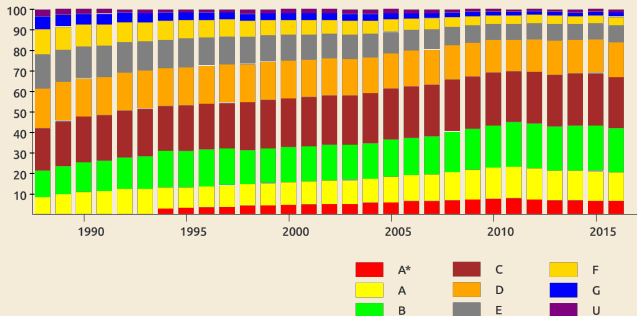
▶ 4 “standard pass”  
before 2019: D–G

▶ 3–1, U “fail”

▶ typically 7–9 subjects for GCSEs

▶  $\geq 5$  GCSEs at pass often the minimum  
including English, maths & science;  
for sixth form can be  $\geq 5$  GCSEs at grade  $\geq 6$

UK GCSE classifications (letter system)



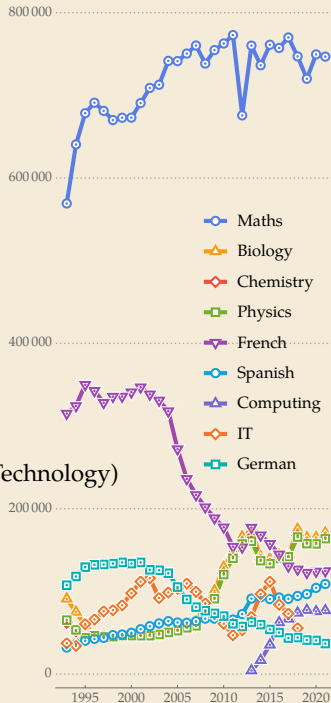
# GCSE subject choice

- ▶ subjects chosen during Year 9 or Year 8
- ▶ Maths, English and Science compulsory at GCSE
  - ▶ “triple science” or not?  
Science can be split (Biology, Chemistry, and Physics)  
or taken as Combined Science
- ▶ typically in addition
  - ▶ a (modern) foreign language
  - ▶ a humanity (e. g., History, Geography, Religious Studies)
  - ▶ an art (Music, Drama, Art, design, Media Studies)
  - ▶ a technical subject (**Computer Science**, Design and Tech, Food Technology)
  - ▶ optionally Physical Education



Computing is a *core subject*

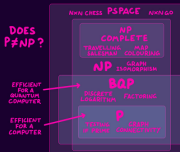
- ↪ compulsory to take throughout KS 4,  
but *not* compulsory *Computer Science* as GCSE



## **1.2 The Role of Computer Science**

# MAP OF COMPUTER SCIENCE

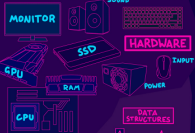
## COMPUTATIONAL COMPLEXITY



## INFORMATION THEORY



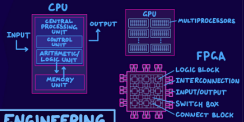
## CRYPTOGRAPHY



## SCHEDULING



## COMPUTER ARCHITECTURE



## THEORETICAL COMPUTER SCIENCE

### COMPUTABILITY THEORY

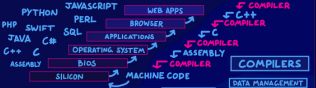


### TURING MACHINE



## COMPUTER ENGINEERING

### SOFTWARE AND PROGRAMMING LANGUAGES



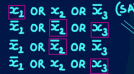
## ALGORITHMS



## OPTIMISATION



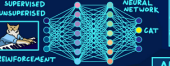
## BOOLEAN SATISFIABILITY



## SUPER COMPUTING



## MACHINE LEARNING



## COMPUTER VISION



## ARTIFICIAL INTELLIGENCE



## ROBOTICS



## NATURAL LANGUAGE PROCESSING



## IMAGE PROCESSING



## KNOWLEDGE REPRESENTATION

## TELEPRESENCE



## AUGMENTED REALITY



## HUMAN COMPUTER INTERACTION



## INTERNET OF THINGS





# Pre-Session Task



**COMPUTING AT SCHOOL**  
EDUCATE · ENGAGE · ENCOURAGE  
In collaboration with BCS, The Chartered Institute for IT

Shaping how our children learn computing

Myths and opportunities


Simon Peyton Jones  
Microsoft Research and Computing at School



- ▶ Simon Peyton Jones' definition of computer science
- ▶ Reasons for teaching CS to every child
- ▶ teaching concepts/ideas as well as skills/technology

# Computing vs. Computer Science vs. ICT/IT vs. Coding vs. ...

- ▶ Definitions of terms: (following Computing at School Guide)
  - ▶ **Computing:** the (entirety of the) subject as laid out in the National Curriculum; the name that will/should appear on timetables etc. Covers *computer science, information technology* and *digital literacy*.
  - ▶ **Computer Science (CS):** scientific and practical study of computation: what can be computed, how to compute it, and how computation may be applied to the solution of problems.
  - ▶ **Information technology (IT):** how computers and telecommunications equipment work, and how they may be applied to the storage, retrieval, transmission and manipulation of data.
  - ▶ **Digital literacy (DL):** ability to effectively, responsibly, safely and critically navigate, evaluate and create digital artefacts using a range of digital technologies.

 Terminology not universally agreed upon. You may find different terms in the literature.

# The National Curriculum in Computing

► <https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study>

Statutory guidance

## National curriculum in England: computing programmes of study

Published 11 September 2013

### Purpose of study

A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world. Computing has deep links with mathematics, science and design and technology, and provides insights into both natural and artificial systems. The core of computing is computer science, in which pupils are taught the principles of information and computation, how digital systems work and how to put this knowledge to use through programming. Building on this knowledge and understanding, pupils are equipped to use information technology to create programs, systems and a range of content. Computing also ensures that pupils become digitally literate – able to use, and express themselves and develop their ideas through, information and communication technology – at a level suitable for the future workplace and as active participants in a digital world.

### Aims

The national curriculum for computing aims to ensure that all pupils:

- can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
- can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
- can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
- are responsible, competent, confident and creative users of information and communication technology

### Attainment targets

By the end of each key stage, pupils are expected to know, apply and understand the matters, skills and processes specified in the relevant programme of study.

Schools are not required by law to teach the example content in [square brackets].

## Subject content

### Key stage 1

Pupils should be taught to:

- understand what algorithms are, how they are implemented as programs on digital devices, and that programs execute by following precise and unambiguous instructions
- create and debug simple programs
- use logical reasoning to predict the behaviour of simple programs
- use technology purposefully to create, organise, store, manipulate and retrieve digital content
- recognise common uses of information technology beyond school
- use technology safely and respectfully, keeping personal information private; identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies

### Key stage 2

Pupils should be taught to:

- design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
- use sequence, selection, and repetition in programs; work with variables and various forms of input and output
- use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs
- understand computer networks, including the internet; how they can provide multiple services, such as the World Wide Web, and the opportunities they offer for communication and collaboration
- use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content
- select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information
- use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact

### Key stage 3

Pupils should be taught to:

- design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem
- use 2 or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions
- understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers [for example, binary addition, and conversion between binary and decimal]
- understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems
- understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits
- undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users
- create, reuse, revise and repurpose digital artefacts for a given audience, with attention to trustworthiness, design and usability
- understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct, and know how to report concerns

### Key stage 4

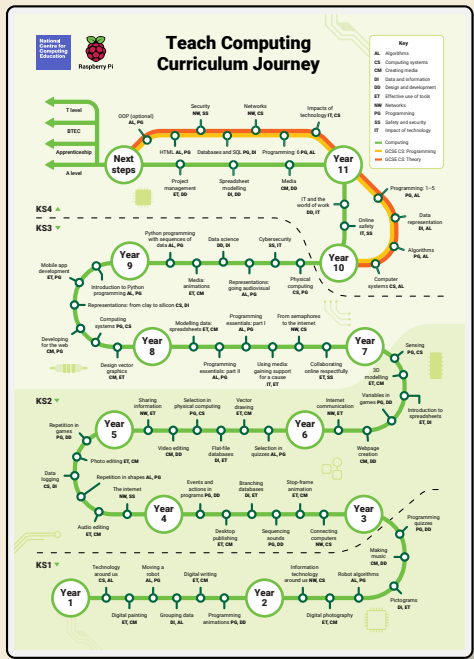
All pupils must have the opportunity to study aspects of information technology and computer science at sufficient depth to allow them to progress to higher levels of study or to a professional career.

All pupils should be taught to:

- develop their capability, creativity and knowledge in computer science, digital media and information technology
- develop and apply their analytic, problem-solving, design, and computational thinking skills
- understand how changes in technology affect safety, including new ways to protect their online privacy and identity, and how to report a range of concerns

# Further Resources

- ▶ National Curriculum *very* abstract and high-level!
  - ▶ Several not-for-profits offer worked out lessons
    - ▶ Teach Computing  
<https://teachcomputing.org/curriculum>  
fully worked out curriculum
    - ▶ Computing at School  
<https://www.computingatschool.org.uk>  
large grassroots collection of resources (≈ forum)
    - ▶ BBC Bitesize revision  
<https://www.bbc.co.uk/bitesize/subjects/zft3d2p>  
overview of topics
    - ▶ Project Quantum  
<https://diagnosticquestions.com/quantum>  
collaborative multiple-choice question bank
- ⇒ good source of inspiration for your lessons!

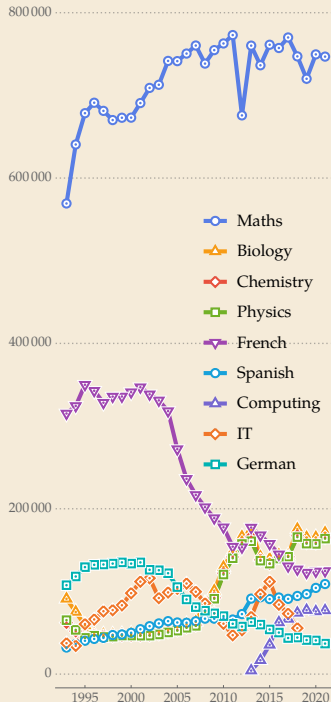


## **1.3 Computing in KS 3**

# Why target KS 3?

- ▶ Key Stage 3 (Years 7–9) particularly influential
  - ▶ “real” programming starts
  - ▶ choice of GCSEs at end of KS 3 often also influence available A-Level choices

*Let's change this picture!*



# Computing Topics in KS 3

- ▶ Programming (loops, conditions, etc.)
- ▶ Data structures (lists, tables, arrays)
- ▶ Abstraction and modeling problems
- ▶ Boolean logic
  - ▶ In code
  - ▶ Circuits, logic gates
- ▶ Assembly language
- ▶ Binary representation
  - ▶ Binary maths
  - ▶ Conversion to/from decimal
- ▶ Unicode and ASCII characters (bit patterns)
- ▶ Hardware components of a PC
- ▶ Software components (operating system)
- ▶ Networking (routers, switches, addresses, etc.)
- ▶ Sorting & searching (algorithms, related to complexity)
- ▶ Algorithmic complexity
- ▶ Graph theory (shortest path, etc.)
- ▶ Cyber security, cookies, hacking, etc.
- ▶ Staying safe online

## Further Resources for KS 3

- ▶ Teach Computing KS3 Curriculum  
<https://teachcomputing.org/curriculum/key-stage-3>
- ▶ Computing at School Guide *Computing in the national curriculum*  
<https://www.computingatschool.org.uk/secondary>
- ▶ BBC bitesize <https://www.bbc.co.uk/bitesize/subjects/zvc9q6f>



## **1.4 Widening Participation**

# Widening Participation

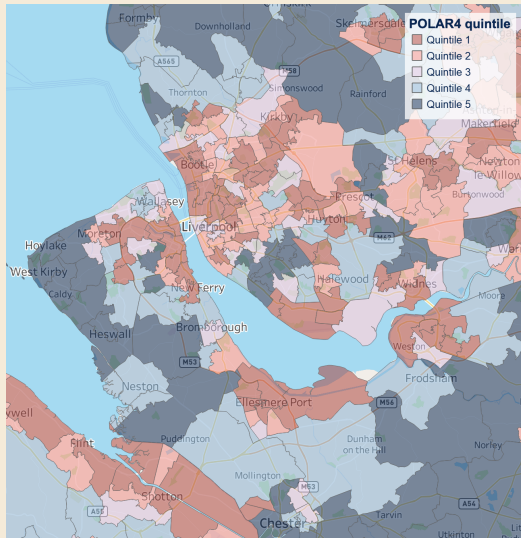
- ▶ The university has a commitment to *widening participation*
  - ▶ Encouraging pupils from disadvantaged backgrounds to study a degree
  - ▶ One of six policy areas of the Russell Group
  - ▶ *Computer Science Taster Days* are part of this initiative
- ▶ There is a central Widening Participation and Outreach team at UoL
  - ▶ Aims to widen participation in our degree programs
  - ▶ Nurtures local talent and potential
  - ▶ Targets pupils who are under-represented in higher education

<https://www.liverpool.ac.uk/widening-participation/>:

*“Our aim is to inspire, engage and enable all those who would not traditionally consider higher education (HE), to fulfil their potential by raising their awareness, challenging barriers and providing opportunities.”*

# Higher Education Participation rates

- ▶ Participation of Local Areas (POLAR4)
  - ▶ Each postcode assigned to a quintile
  - ▶ Q1 is fewest HE students
- ▶ Across Liverpool: low participation rate in higher education
- ↪ Need to make Taster Days engaging
  - ▶ University as an achievable goal
  - ▶ Be honest without being elitist
- ↪ *You have the opportunity to make your lesson have a big impact on students*



<https://www.officeforstudents.org.uk/data-and-analysis/young-participation-by-area>

# Implications for your lesson plan

- ▶ Main purposes of Taster Day lessons
  - ▶ support teachers in delivering the National Curriculum
  - ▶ show that university life is not elitist, but open to all backgrounds
- ▶ Each class will contain a wide range of abilities and motivation
  - ▶ Not everyone will be as engaged as you were at that age
  - ▶ May have no serious ambition to go to university at all
- ▶ Computing and technology are pervasive in all aspects of modern life
  - ▶ Obviously lot of background theory that we can't ignore
  - ▶ Cannot “dumb down” CS

↪ Good lessons are

- ▶ *relatable*: using familiar concepts
- ▶ *relevant*: tied to students' experience/environment
- ▶ *achievable*: plan success in activities/tasks, maybe at different levels