COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE

BODY OF KNOWLEDGE

KEY STAGES 1, 2, 3, 4 AND 5

Draft version 6

Editor: Dr K R Bond

With contributions from the BOK sub-committee:

Jack Lang

Adrian Jackson

John Woollard

And contributions from CAS Working Members team:

Emma Wright

Andrew Herbert

Simon Peyton-Jones

Chris Bishop

Roger Boyle

Miles Berry

Simon Humphreys

Aaron Sloman

Jonathan Stout

Roger Davies

Shahneila Saeed

Steve Hunt

Mandy

Table of Contents

Table of Contents

Introduction

Computing is a science of information processes. The traditional way of focusing on ideas behind computing technologies puts the computer, rather than the computation, at the centre. Traditional courses have tended to focus on the computer as a tool at the expense of revealing the central concepts of the discipline.

Establishing a Body of Knowledge (BOK) for Computing across the National Curriculum (NC) Key Stages 2, 3, 4 and 5 is no mean undertaking. We have tried to avoid falling into the trap of arguing on the basis of intuition and anecdote because such an approach cannot easily change the mind of someone with a different intuition or anecdote [Lister 2007[1]]. Instead, we have tried to adopt an evidenced-based approach. Inevitably, this has been a very time-consuming task but some positives have emerged. Several exemplar curricula exist already - Israel, South Africa, ACM's K12 and Bermuda (created by Stanford university, US). These were considered as potential candidates. Much good work has also been done towards providing appropriate exemplifying material to aid delivery of a Computing curriculum for the age ranges covered by NC Key Stages 2, 3, 4 and 5 - CSInside, CS4FN, CSUnplugged, University of Washington Benefit, Fluency with IT, (http://courses.washington.edu/benefit/FIT100/), Centre For Innovation in learning, Virginia Tech (http://courses.cs.vt.edu/~csonline/).

Research programmes aimed at establishing appropriate and effective teaching material for the different age ranges have been funded. It is clear from the evidence reported in the literature that students engage enthusiastically with carefully designed exercises that do not involve programming a computer as much as or more than exercises that do. On the competition front some Informatics Olympiads have recognised this by offering both non-programming based competitions as well as programming-based competitions [Burton 2008[2], Dagiene 2008[3]]. Most students are motivated by puzzle-solving activities but a difficulty arises when this activity involves programming as well. "In a mathematics contest, any student can follow their nose and scribble ideas down. In a programming contest - certainly the traditional type in which programs are scored according to their behaviour - a student cannot score any points (or even have their submissions judged) unless they can create a running program in a relatively short period of time."[Burton 2008]. Instead of motivating the desire to learn, coupling problem solving with programming can have the opposite effect [Burton 2008]. However, it is acknowledged that for some students programming a computer can be highly motivational. All generalisations are dangerous including this one.

[1] Lister 2007, Computer Science Teachers as Amateurs, Students and Researchers Raymond Lister, University of Technology, Sydney, Faculty of Information Technology, Broadway, NSW 2007, Australia, Koli Calling 2005 Conference on Computer Science Education, held in November 2005]

[2] Burton 2008, Thirty-First Australasian Computer Science Conference (ACSC2008), Wollongong, Australia. Conferences in Research and Practice in Information Technology

[3] Dagiene 2008 Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks. Valentina Dagiene and Gerald Futschek 2008

The Australian Informatics Olympiad committee and the organising committee of the Bebras Informatics Contest have adopted an approach that attempts to encourage the teaching of Informatics in an attractive way with the emphasis being on developing thinking skills not mastery of a programming language. The following countries are currently participating in the International Bebras contest: Denmark, Estonia, Finland, Germany, Latvia, Sweden, and Poland, (Austria, Egypt, Israel and The Netherlands. Over 40000 students entered for this competition in 2007. The literature is full of discussion on the teaching of programming and programming languages. No clear picture has emerged especially regarding the teaching of novice programmers or how to deal with the lack of success in teaching novice programmers. This is not surprising when factors such as the learning styles of students, their prior knowledge and their stages of intellectual development are taken into account.

Without doubt, there are students who display a natural gift for programming but developing a body of knowledge just for these students would not be right. Lack of an optimal solution for teaching programming seems to have encouraged a folk-lore approach which may be why we have ended up in the current position.

Concrete to Abstract

From a survey of the literature, it is quite clear that how computer science is taught is equally as important as what is taught. The message is that pedagogy matters. This viewpoint is reinforced by The American Association for the Advancement of Science for teaching science in its highly praised publication Science for All Americans. This publication recommends that students' learning progression should follow a pathway from the concrete to the abstract. Quoting from this publication,

> "Young people usually learn most readily about things that are tangible and directly accessible to their senses. With experience, students develop an ability to understand abstract concepts, manipulate symbols, reason logically and generalise but it should be recognised that these skills develop slowly. In fact, the dependence on concrete examples persists throughout life for many people. Concrete experiences are most effective in learning when they occur in the context of some relevant conceptual structure."

The latter point is a very important one. In general, concrete learning experiences divorced from relevant conceptual structures provide little insight into a subject as a whole.

The literature raises three issues with regard to computer science teaching at secondary and primary levels:

1. Which of the many concepts of computer science should be taught
2. What contexts should be used to deliver these concepts
3. How should the movement from the concrete to the abstract be supported

In science teaching concepts abound. Science curricula emphasise the aspects of science that young people must learn and know. Is there an equivalent in computer science? Jeanette Wing[4], in her manifesto, Computational Thinking, emphasises the ubiquity of computation. She argues for the adoption of computational thinking as a fourth "analytical ability" alongside reading, writing and arithmetic.

---

[4] Wing 2006, Computational Thinking, Commun. ACM 49, 3 (Mar. 2006), 33-35

Context helps to communicate the "magic and beauty" of the discipline of computer science Context can range from multimedia work through robotics, animation, simulation, storytelling, e-textiles and game authorship using Alice (in 3D), Scratch (in 2D) and BYOB, Lego Mindstorms, Greenfoot (2D), Jeroo, Processing and the LilyPad Arduino.

The discipline of computer science has abstraction at its core. Computer scientists think at multiple levels of abstraction [Wing 2006[5]]. However, an approach should be used involving a teaching and learning strategy that begins with the concrete in a context familiar to students and then gradually leads to an understanding of the abstract. Identifying a strategy that achieves this is the most challenging of the three issues.

Design of Computer Science Curriculum

A common thread of the research literature suggests strongly that the design of computer science curricula for 5 - 18 (CSC5-18) should rely on:

- Central concepts of the discipline
- Not on technical short-term developments

[5] Wing 2006, Computational Thinking, Commun. ACM 49, 3 (Mar. 2006), 33-35

Overview of a curriculum for 5 - 18

Age	Stage	Computing
3	Nursery (non-compulsory)	
4 - 5	**Primary – Key Stage 1** Reception class	Acquire confidence and pleasure playing with a computer and exploring its potential.
5 - 6	Year 1	
6 - 7	Year 2	
7 - 8	**Key Stage 2** Year 3	• Learn a programming language; construct and run simple programs. There are several languages designed for small children. • Understand the use of hierarchy for the organisation of files, emails, programs, etc. • Understand the concept of software bug, as opposed to a fault in the hardware. • Understand the pervasive nature of computing embedded in everyday devices: ○ mobile phones, smart cards, washing machines, digital TV and radio, cars, etc. • Experience how computers can control devices, e.g., a mobile robot, a light show.
8 - 9	Year 4	
9 - 10	Year 5	
10 - 11	Year 6	
11 - 12	**Secondary – Key Stage 3** Year 7	• Understand the need for computer security and approaches to improving it such as good passwords, encryption, security protocols, etc. • Understand how functionality can be built up in layers from the hardware to the application via abstraction and virtual machines. • Understand how instructions at a higher layer can be compiled or interpreted into a lower layer. • Understand how information at a higher layer can be encoded into a lower layer, e.g. storing images. • Understand how computer programs can be used to model natural, artificial and unreal situations. Experience controlling a simulation. • Understand the structure of the Internet both at hardware and software levels. • Understand that information can be structured and unstructured. Experience use of html and css to structure information • Understand that structuring of information is useful when searching. Experience searching the Web efficiently and searching a database using SQL • Understand the difference between closed systems and open systems programming and experience both, e.g. spreadsheet functions, games programming • Experience how computers can control devices with a range of sensors and actuators, e.g. mobile robot, a light show
12 - 13	Year 8	
13 - 14	Year 9	

14 - 15	**Key Stage 4** Year 10	• Understand and experience the interactive world and the mathematical world view of computing via event-driven, GUI and non-event driven programming • Understand the concept of an algorithm through examples such as sorting, searching, etc; and that one algorithm can be more efficient than another. • Understand how information at a higher layer can be encoded into a lower layer in compressed and uncompressed forms, with and without error, e.g. storing numbers with a fractional part. • Understand the internal architecture and operation of a digital computer from logic gate level upwards • Understand how to digitise information • Understand techniques for encoding, encrypting, sending receiving, detecting errors in messages • Extend understanding of how computer programs can be used to model natural, artificial and unreal situations. Experience creating models that support simulations of natural or artificial or unreal worlds, e.g. ant colony simulation in StarLogo TNG • Understand how to create and query tables in a database using DDL and DML • Understand how information can be organised to enable fast retrieval, the use of search engines • Understand the need for unique hierarchical naming systems, e.g. Internet namespace, and levels of abstractions, names, handles, addresses, locations • Understand the principle of locality in the context of programming and caching of information • Understand the concept of thrashing • Understand the different types of storage systems and their organisation • Understand and experience programming client-server operation and its application to networking. • Understand and experience the use of abstraction, decomposition and information hiding in problem solving • Understand the need for coordination • Program robots to understand key issues with automation, strong and weak AI • Debate the philosophical issues involved in artificial intelligence.
15 - 16	Year 11	
	End of Compulsory Schooling	
16 - 17	Key Stage 5 Year 12	See current AS specifications
17 - 18	Year 13	See current A2 specifications

Body of Knowledge Framework

Body of Knowledge Framework

Research suggests that the most appropriate framework for a Computer Science body of knowledge is one that combines structure and process as learning content to achieve a harmonious balance of fundamental principles and practice.

Structure as learning content = concepts, fundamental ideas and principles

- This means addressing in the teaching objectives the following:
    1. Basic concepts
    2. Categories
    3. Principles

Process as learning content = developing thinking skills for, acquiring knowledge and understanding of the following

1. Problem solving and problem posing
2. Classifying
3. Finding relationships
4. Investigating
5. Analysing
6. Generalising
7. Communicating
8. Questioning
9. Ordering
10. Comparing

A principles-based approach is common in other fields. Computer Science has now reached a level of maturity where it is possible to articulate the field in terms of fundamental principles, but what principles? To answer this question we have turned to the Great Principles Project- [6]http://greatprinciples.org.

John Maeda[7] of MIT responds to people who ask "What programming language should be taught?".

John's stock reply is "What principle are you trying to teach?

The design of computer science curricula for 5 – 18 should rely on

- **Central concepts of the discipline**
- **Not on technical short-term developments**

6 I am indebted to Peter Denning for giving his permission to use material from the Great Principles project - Dr K R Bond.

[7] Design by Numbers, MIT Press, ISBN-13: 978-0-262-13354-8

Principles Framework

The framework developed has been based upon Peter Denning's Seven Principles and Four Core Practices:

Principles:

1. Computation
2. Communication
3. Coordination
4. Recollection
5. Automation
6. Evaluation
7. Design

Core practices:

1. Programming (including multilingual programming practice)
2. Systems and systems thinking
3. Modelling, validating, testing, and measuring
4. Innovating

Embedded within these are central processes for computer science education of problem solving and problem posing, classifying, finding relationships, investigating, analysing, generalising and central processes for applied computer science education of communicating, questioning, ordering and comparing.

Descriptions of the seven principles as given by Peter Denning are summarised in Table 1.

Principle	**Summary**
Computation	These principles address what processes, natural and artificial, are computational, what they can and cannot do, and how we cope with inherent and pervasive computational complexity.
Communication	These principles concern the transmission of data with reliable reception.
Coordination	These principles concern how autonomous entities work together toward a common result.
Recollection	These principles concern how computations store and recall information, and how data layout in the storage system affects their performance.
Automation	These principles concern finding efficient computational ways to perform human tasks. Tasks can be mental, such as doing arithmetic, playing chess, and planning schedules, or physical, such as running an assembly line, driving a car, controlling an airplane.
Evaluation	These principles concern how computing systems perform under various computational loads and how much capacity they need to deliver their results on time.
Design	These principles concern how to design software and computing systems that are dependable, reliable, usable, safe, and secure (DRUSS).

Table 1 The seven principles from the Great Principles Project

Each of the seven principles is supported by a principle's narrative that attempts to explain why each is a fundamental principle - http://cs.gmu.edu/cne/pjd/GP/gp_narratives.html.

The seven principles in Table 1 were used for the framework of CAS's body of knowledge and were expressed more concisely for this framework as follows:

- Computation
  - Meaning and limits of computation
- Communication
  - Reliable data transmission
- Coordination
  - Cooperation among networked entities
- Recollection
  - Storage and retrieval of data
- Automation
  - Discovering efficient computational ways to perform human tasks
- Evaluation
  - Performance, prediction and capacity planning
- Design
  - Building reliable software systems

A principles-oriented descriptive framework should:

- Reveal Computer Science's deep structures
- How they apply in many fields
- Reveal common aspects of technologies
- Create opportunities for innovation
- Open entirely new ways to stimulate the excitement and curiosity of young people about the world of computing

The principles-based framework was expressed in the following conceptual groups for convenience:

1. Data and information
2. Computation
3. Communication and storage
4. Systems
5. Design
6. Coordination
7. Automation

Conceptual Groups

1. **Data and information**
    1. REPRESENTATIONS HOLD INFORMATION
    2. REPRESENTATIONS CAN BE COMPRESSED
    3. FINITE REPRESENTATIONS OF REAL PROCESSES ALWAYS CONTAIN ERRORS
    4. INFORMATION RETRIEVAL

2. **Computation**
    1. COMPUTATIONS CAN BE OPEN OR CLOSED
    2. COMPUTATION IS A SEQUENCE OF REPRESENTATIONS
    3. COMPUTATIONS HAVE CHARACTERISTIC SPEEDS OF RESOLUTION
    4. COMPLEXITY MEASURES THE TIME OR SPACE ESSENTIAL TO COMPLETE COMPUTATIONS

3. **Communication and storage**
    1. INFORMATION CAN BE ENCODED INTO MESSAGES
    2. DATA COMMUNICATION ALWAYS TAKES PLACE IN A SYSTEM CONSISTING OF A MESSAGE SOURCE, AN ENCODER, A CHANNEL, AND A DECODER
    3. MESSAGES CORRUPTED DURING TRANSMISSION CAN BE RECOVERED
    4. MESSAGES CAN BE ENCRYPTED
    5. MESSAGES CAN BE COMPRESSED
    6. HIERARCHICAL NAMING SYSTEMS ALLOW LOCAL AUTHORITIES TO ASSIGN NAMES THAT ARE GLOBALLY UNIQUE IN VERY LARGE NAME SPACES
    7. ACCESS TO STORED OBJECTS IS CONTROLLED BY DYNAMIC BINDINGS BETWEEN NAMES, HANDLES, ADDRESSES AND LOCATIONS
    8. DATA CAN BE RETRIEVED BY NAME OR BY CONTENT
    9. THE PRINCIPLE OF LOCALITY DYNAMICALLY IDENTIFIES THE MOST USEFUL DATA, WHICH CAN THEN BE CACHED AT THE TOP OF THE HIERARCHY
    10. THRASHING IS A SEVERE PERFORMANCE DEGRADATION CAUSED WHEN PARALLEL COMPUTATIONS OVERLOADTHE STORAGE SYSTEM
    11. ALL COMMUNICATIONS TAKE PLACE IN STORAGE SYSTEMS
    12. STORAGE SYSTEMS COMPRISE HIERARCHIES WITH VOLATILE (FAST) STORAGE AT THE TOP AND PERSISTENT (SLOWER) STORAGE AT THE BOTTOM

4. **Systems**
    1. SYSTEMS THINKING

5. **Design**
    1. ABSTRACTION, INFORMATION HIDING AND DECOMPOSITION ARE COMPLEMENTARY ASPECTS OF MODULARITY
    2. THE FOUR BASE PRINCIPLES OF SOFTWARE DESIGN ARE HIERARCHICAL AGGREGATION, LEVELS, VIRTUAL MACHINES, AND OBJECTS.
    3. DESIGN PRINCIPLES ARE CONVENTIONS FOR PLANNING AND BUILDING CORRECT, FAST, FAULT TOLERANT, AND FIT SOFTWARE SYSTEMS.
    4. OBJECTS ORGANIZE SOFTWARE INTO NETWORKS OF SHARED ENTITIES THAT ACTIVATE OPERATIONS IN EACH OTHER BY EXCHANGING SIGNALS.
    5. IN A DISTRIBUTED SYSTEM, IT IS MORE EFFICIENT TO IMPLEMENT A FUNCTION IN THE COMMUNICATING APPLICATIONS THAN IN THE NETWORK ITSELF (END-TO-END PRINCIPLE).

6. **Coordination**
    1. A COORDINATION SYSTEM IS A SET OF AGENTS INTERACTING WITHIN A FINITE OR AN INFINITE GAME TOWARD A COMMON OBJECTIVE
    2. ESSENTIAL ELEMENTS OF COORDINATION SYSTEMS

3. THE PROTOCOLS OF COORDINATION SYSTEMS MANAGE DEPENDENCIES OF FLOW, SHARING AND FIT AMONG ACTIVITIES
4. COORDINATION TASKS CAN BE DELEGATED TO COMPUTATIONAL PROCESSES
5. ACTION LOOP IS THE FOUNDATIONAL ELEMENT OF ALL COORDINATION PROTOCOLS

7. Automation

THESE PRINCIPLES CONCERN FINDING EFFICIENT COMPUTATIONAL WAYS TO PERFORM HUMAN TASKS. TASKS CAN BE PHYSICAL, SUCH AS RUNNING AN ASSEMBLY LINE, DRIVING A CAR, CONTROLLING AIRPLANE SURFACES; OR MENTAL, SUCH AS DOING ARITHMETIC, PLAYING CHESS, AND PLANNING SCHEDULES.

1. PHYSICAL AUTOMATION MAPS HARD COMPUTATIONAL TASKS TO PHYSICAL SYSTEMS THAT PERFORM THEM ACCEPTABLY WELL.

Learning Content

For the 11 - 18 years age range, the learning content of courses based on the principles-based framework should cover a selection of the following to a lesser or greater degree:

- Programs: how they are written, represented as data, parsed, transformed, compiled or interpreted, and run.
- Programming languages: imperative, event-driven, object-oriented, functional, logic, assembly.
- Algorithms and their structure: sequencing, branching, looping, recursion, sub-routines.
- Complexity and efficiency in both time and space.
- Computer architectures: the Von Neumann and other machine designs.
- Operating systems.
- Abstraction, virtual machines and their application programming interfaces.
- Interactive/reactive computing
- Embedded systems and pervasive computing.
- User interfaces.
- Software engineering and the construction of dependable complex systems.
- Computational models.
- Artificial intelligence and cognitive science.

Curriculum time for Computing varies from centre to centre. It was felt that a body of knowledge expressed at the level of a generalised framework, i.e. an overview, so typical of many recent educational initiatives would not assist centres to fully grasp the potential of a principles-based curriculum. Nor would such a generalised framework clearly signpost the differences between the abandoned computing 11-16 curriculum of the eighties with its emphasis on programming, its factory-model of education approach and the 5-18 Computing curriculum that we wish to see adopted for the twenty-tens based on an artist's studio model not a mass-education factory model. A principles-specific framework expressed with sufficient depth and clarity aids articulation and justification of a curriculum that can be customised to suit the individual needs of students through flexibility in the range of actual content, its depth, form of expression and timing. Principles can be covered in a multitude of ways, implicitly or explicitly for the student. With a principles-based framework a teacher should always be able to articulate in terms of principles and concepts why students are engaging in, collaboratively or otherwise, particular exercises, discussions and projects. A principles-based framework supports a student-centric model of learning. Furthermore, an approach based on carefully articulated principles and concepts clearly makes a strong case for treating Computing in schools as a first class academic discipline supporting other academic disciplines as well as technology.
The nineteenth century chemists whose pioneering scientific work led to the creation of the chemical industry had little difficulty viewing chemistry as both an academic discipline supporting other academic disciplines as well as the industrial technology of the day. No self-respecting professional chemist of that era would have been without excellent glass-blowing skills. The difference between the glass-blowers of Venice and one of these chemists lay in the knowledge and understanding possessed by the latter of the fundamental principles and concepts of chemistry. It is the latter that anticipates and is instrumental in changing perceptions and ultimately is responsible for driving changes

in technology. The physicist Denis Gabor mapped out the theory of holography long before it became possible to make holograms.

It is not expected that the principles-based body of knowledge framework be covered in its entirety in any 9-18 curriculum but that its use can inform the development of tailored curricula for particular needs and interests. The one-size fits all model is anathema.

Criteria for Fundamental Ideas

Four criteria were used to help to identify fundamental ideas of the computing body of knowledge:

1. Horizontal criterion
2. Vertical criterion
3. Criterion of time
4. Criterion of sense

For each of the above:

- Horizontal criterion:
  - the fundamental idea is applicable or observable in multiple ways in different areas of computing
- Vertical criterion:
  - the fundamental idea may be demonstrated and taught at every intellectual level from kindergarten up to university level
- Criterion of time:
  - the fundamental ideas taught should be of long lasting relevance

Criterion of sense:

- the fundamental idea is related to everyday language and/or thinking

By a principle is meant a statement that guides or constrains future action.

Computing principles are of two kinds:

1. Recurrences, including laws, processes, and methods that describe repeatable cause-effect relationships
   – An example of a law is that the fastest sorting algorithms take time at least order of n log n to arrange n items in order
2. Guidelines for conduct
   – An example of a conduct guideline is that network programmers should divide protocol software into layers
   – The purpose of such principles is to reduce apparent complexity, increase understanding, and enable good design

Aims

- To achieve a balance between

  –practice and principle

- Computing knowledge space:

  1. Principles
  2. Practice

- Both equally important

LEARNING OUTCOME STANDARDS

KS	1. Data & Information	2. Computation	3. Communications & Storage	4. Systems	5. Design	6. Coordination	7. Automation

Content Learning Standards

KS	1. Data & Information	2. Computation	3. Communications & Storage	4. Systems	5. Design	6. Coordination	7. Automation
1							
2	1.1.1 – 1.1.7 1.4.1	2.1.3	3.1.1 3.5.2 3.6.1-3.6.2 3.6.5 3.7.1 3.7.4 – 3.7.6 3.7.9 3.7.13 3.8.1 3.8.3 - 3.8.4 (parts) 3.11.1 – 3.11.2 (parts) 3.11.6 3.12.1		5.4.1 5.4.2	6.1.2 6.1.6 6.1.7 6.3.2 6.3.3 6.4.1(part)	7.1.2 7.1.4

KS	1. Data & Information	2. Computation	3. Communications & Storage	4. Systems	5. Design	6. Coordination	7. Automation
			3.12.3 4.1.1				
3	1.1.1-1.1.12 1.1.14-1.1.16 1.2.1 – 1.2.2 1.2.4 – 1.2.5 1.4.1	2.1.2 – 2.1.4 2.2.1 – 2.2.10 2.3.1 – 2.3.5 2.4.1 – 2.4.3	3.1.1 – 3.1.5 3.2.1 – 3.2.2 3.2.5 3.2.7 3.3.1 – 3.3.2 3.4.2 – 3.4.3 3.5.2 3.6.1 – 3.6.2 3.6.5 – 3.6.6 3.7.1 3.7.4 – 3.7.14 3.8.1 3.8.3 – 3.8.4 3.9.1 – 3.9.4 3.10.1 – 3.10.2 3.11.1 – 3.11.8 3.12.1 – 3.12.3	4.1.1 – 4.1.2	5.1.2 – 5.1.3 5.2.1 – 5.2.2 5.3.1 5.3.4 5.3.7 5.4.1 - 5.4.2	6.1.1 – 6.1.8 6.2.1 – 6.2.9 6.3.1 – 6.3.5 6.4.1 6.5.1	7.1.2 7.1.4

KS	1. Data & Information	2. Computation	3. Communications & Storage	4. Systems	5. Design	6. Coordination	7. Automation
4	1.1.1 – 1.1.12 1.1.14 – 1.1.16 1.2.1 – 1.2.2 1.2.4 – 1.2.6 1.3.1 1.4.1	2.1.2 – 2.1.4 2.2.1 – 2.2.11 2.3.1 – 2.3.5 2.4.1 – 2.4.3 2.4.5	3.1.1 – 3.1.5 3.2.2 3.2.5 3.3.1 – 3.3.2 3.4.2 – 3.4.3 3.5.2 3.5.5 3.6.1 – 3.6.2 3.6.5 – 3.6.6 3.7.2 – 3.7.10 3.7.12 – 3.7.14 3.8.1 – 3.8.4 3.9.1 – 3.9.4 3.10.1 – 3.10.2 3.11.1 – 3.11.8 3.12.1 – 3.12.3	4.1.1 – 4.1.2	5.1.1 – 5.1.4 5.2.1 – 5.2.4 5.3.1 – 5.3.2 5.3.4 5.3.6 – 5.3.7 5.4.1 – 5.4.2 5.5.1	6.1.1 – 6.1.8 6.2.1 – 6.2.7 6.2.9 6.3.1 – 6.3.5 6.4.1 6.5.1	7.1.2 7.1.4
5	1.1.1 – 1.1.16	2.1.1 – 2.1.4	3.1.1 – 3.1.5	4.1.1 –	5.1.1 –	6.1.1 – 6.1.8	7.1.1 – 7.1.4

KS	1. Data & Information	2. Computation	3. Communications & Storage	4. Systems	5. Design	6. Coordination	7. Automation
	1.2.1 – 1.2.6 1.3.1 – 1.3.2 1.4.1	2.2.1 – 2.2.12 2.3.1 – 2.3.5 2.4.1 – 2.4.10	3.2.2 3.2.5 3.2.7 3.3.1 – 3.3.2 3.4.2 – 3.4.3 3.5.1 – 3.5.2 3.5.4 – 3.5.7 3.6.1 – 3.6.6 3.7.2 – 3.7.14 3.8.1 – 3.8.4 3.9.1 – 3.9.6 3.10.1 – 3.10.2 3.11.1 – 3.11.8 3.12.1 – 3.12.3	4.1.2	5.1.4 5.2.1 – 5.2.4 5.3.1 – 5.3.7 5.4.1 – 5.4.4 5.5.1	6.2.1 – 6.2.9 6.3.1 – 6.3.5 6.4.1 6.5.1	

KS	Problem solving and problem posing	Classifying	Finding relationships	Investigating	Analysing	Generalising	Communicating	Questioning	Ordering	Comparing

Process Learning Standards

KS	Problem solving and problem posing	Classifying	Finding relationships	Investigating	Analysing	Generalising	Communicating	Questioning	Ordering	Comparing
1		Yes					Yes		Yes	Yes
2		Yes	Yes	Yes			Yes		Yes	Yes
3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Principles for Key Stages 3, 4 and 5

Learning Outcomes

Learning outcomes for Key Stages 3 to 5 Computing. Strand: DATA and INFORMATION

Learners should know and understand……

Principle	Key Stage 3	Key Stage 4	Key Stage 5
1. DATA and INFORMATION PRINCIPLES			
1.1 REPRESENTATIONS HOLD INFORMATION			
1.1.1 A representation is a pattern of symbols that conveys information, e.g. a pattern of 1s and 0s. ***A single bit can represent a state: Lamp On - 1, Lamp Off - 0*** ***A pattern of bits can represent:*** ***Sound samples of actual sounds*** ***Bitmap representation of an image*** ***Text*** ***Program form of***	Bitmap representation of images Resolution of a bitmap representation of an actual image Colour depth of a bitmap representation of an actual image Character encoding of text - ASCII Encoding of unsigned integers for simple arithmetic	Representation of signed integer numbers (a) sign & magnitude (b) 2's complement Fixed point representation of (a) unsigned numbers (b) signed numbers Character encoding of text - ASCII - UNICODE	Floating-point representation (a) unsigned numbers (b) signed numbers Range of (a) fixed-point representation (b) floating-point representation Precision of (a) fixed-point representation (b) floating-point representation

Principle	Key Stage 3	Key Stage 4	Key Stage 5
an algorithm ***In digital computers data are arrangements of bits (0 and 1).*** ***Data are not the only representations in a computer:*** ***The program that controls the computer is also a representation -*** ***It represents an algorithm, a set of mechanical steps that transform input data to output data.***	Machine code form of a computer program		
1.1.2. Symbols can be encoded with patterns of bits.	See above	See above	See above
1.1.3. A computer is a device that transforms data representations under the control of a procedural representation. ***There are only two essentials of computation:*** ***1. A series of representations*** ***2. A set of rules for transforming each representation to the next in the series*** ***The digital computer is not essential.*** ***The computer is one***	Machine code	Machine code Instruction set (a) Opcode (b) Operand Assembly language 3rd generation programming language	Simple machine code operations Equivalent assembly language 3rd generation programming language Finite State Machines Turing machine DNA computing

Principle	Key Stage 3	Key Stage 4	Key Stage 5
of many possible media in which computations can happen. Computing is pervasive because representations are pervasive.			
1.1.4. Meaning is discerned and acted upon by observers reading pattern	To a user the data stand for (hold information about) objects or effects in the real world, e.g. a person's bank account. Sound samples stand for the actual sound. Data are an arrangement of bits. Your brain supplies the meaning when perceiving the arrangement. Unsigned integers Signed integers The bits themselves have no meaning. They do not hold information. Knowledge is required to interpret a bit pattern in order for information to be revealed. Computers act on representations. The computer's output representation can be converted to action by a device that translates representation into a physical effect, such as an image on a VDU.	Information is a brain response to the stimulation of a pattern of bits. Difference between knowledge and information, e.g. knowledge of the English language enables a piece of text written in English to be read and understood thus communicating any information contained within the text. Acquisition of language. Unsigned fixed point numbers Signed fixed point numbers Holding information is a metaphor. Because we live in communities wherein we all assign the same meanings to the same patterns, this metaphor does not render meanings of representations purely subjective.	Information is a brain response to the stimulation of a pattern of bits. Unsigned floating point numbers Signed floating point numbers
1.1.5. Every representation is embodied into physical phenomena	Ink on paper Punched holes in cards Magnetic patterns on a disk surface Bumps and pits on a compact disk	Ink on paper Punched holes in cards Magnetic patterns on a disk surface	Ink on paper Punched holes in cards Magnetic patterns on a disk surface

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	3D folded sequences of amino acids in DNA	Bumps and pits on a compact disk 3D folded sequences of amino acids in DNA	Bumps and pits on a compact disk 3D folded sequences of amino acids in DNA
1.1.6. Continuous representations such as voltages and currents can be represented by patterns of bits because any value of the representation function can be approximated by a binary number.	Analogue to digital conversion Digital to analogue conversion Digitising speech and music Sensors for sensing physical quantities and controlling robot	Analogue to digital conversion Digital to analogue conversion Digitising speech and music Sensors for sensing physical quantities and controlling robot	Analogue to digital conversion Digital to analogue conversion Digitising speech and music Sensors for sensing physical quantities and controlling robot. PCM encoding, sampling rate, sampling resolution, quantisation noise, Nyquist's theorem
1.1.7. Meaning is discerned and acted upon by observers reading pattern	Concept of a data type Simple data types - Integer/Whole number - Number with fractional part - Boolean - Character Structured data types - Strings - one-dimensional arrays	Concept of a data type: – Defines interpretation – Defines amount of storage – Defines operations allowed Simple data types - Integer/Whole number - Number with fractional part (fixed and floating point) - Boolean - Character - Sub-range Structured data types - Strings - One and two-dimensional arrays	Simple data types - Integer/Whole number - Number with fractional part (fixed and floating point) - Boolean - Character - Sub-range - Enumerated type Structured data types - arrays (multi-dimensional) - records - file of any simple of structured type - text files - Set Character encoding

Principle	Key Stage 3	Key Stage 4	Key Stage 5
		- File of byte - File of integer Semi-structured/unstructured data type - Text file Character encoding - ASCII - Unicode Program as data - Interpreters - Viruses - Java applet	- ASCII - Unicode Stack Queues - linear, circular, priority Binary trees Heap Graphs - directed and undirected - adjacency matrix - adjacency list List Linked list Hash table/file Sequential files Random access files
1.1.8. Interpretation of a representation depends on the means by which observers interact with the representation	Interpretation of sensor information Use in robotics, control and measurement	Interpretation of sensor information Use in robotics, control and measurement	An observer of a linear string can parse it into a tree of embedded phrases; the tree is the string's meaning. The physical embodiment affects what information can be derived from it. Parsing a programming language statement to create a parse tree Parsing HTML Regular expressions BNF

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			Reverse Polish Notation
1.1.9. An algorithm is a representation of a method to accomplish a task or process. An algorithm transforms input data to output data in a finite amount of time. Input data are states of an input representation which can be thought of as a set of symbols that represent values. Output data are likewise the states of an output representation. A program is an algorithm plus its data.	Input - list of items of different sizes Output - list of items in some order based on size of item Selection sort, Quicksort (CS Unplugged video) via weighing scales. Comparison of these two sorts Sorting network (CS Unplugged video) Input - list of ordered items Output - the sought item or not found Searching - Linear search (ordered list) - Binary search	Non-recursive programming of Finding primes - Brute force - Using only primes as divisors - Array method based on Sieve of Eratosthenes Finding factors Number conversion algorithms - Denary to binary	Recursive and non-recursive Linear search Binary search Bubble sort Insertion sort Quicksort Tree traversals - inorder - postorder - preorder Graph algorithms - Breadth-first search - Depth-first search Linear queue and circular queue operations Stack operations List operations Finding primes - Brute force - Using only primes as divisors - Array method based on Sieve of Eratosthenes

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			Finding factors Number conversion algorithms - Denary to binary - Denary to hexadecimal - Binary to decimal Calculating square roots - Newton's method
1.1.10. A compiler translates a program to machine code, the low-level bit patterns that drive a machine. An algorithm can be regarded as a logical machine because of the equivalence between a high-level language program and its machine code representation as produced by a compiler.	Compilers	Compilers and linkers Static linking and dynamic linking	Compilers and linkers Static linking and dynamic linking
1.1.11. Rules describe the allowable patterns of carrier configurations	We can tell if a string of symbols belongs to a computer program by checking whether the string of symbols can be parsed by the language's grammar.	We can tell if a string of symbols belongs to a computer program by checking whether the string of symbols can be parsed by the language's grammar. Compiling a program - syntax errors	We can tell if a string of symbols belongs to a computer program by checking whether the string of symbols can be parsed by the language's grammar. Specifying grammar of a language using - Recursive definitions

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			▫ BNF - Syntax diagrams
1.1.12. Representations are finite	The space used for storing values (i.e. their memory) is divided into discrete chunks whose dimensions in each direction are always greater than some positive minimum length (necessarily non-zero). They store one bit of information in each chunk. This means that computers can't store an infinite amount of information in a finite volume of space. Storing bitmaps in compressed and uncompressed form.	The space used for storing values (i.e. their memory) is divided into discrete chunks whose dimensions in each direction are always greater than some positive minimum length (necessarily non-zero). They store one bit of information in each chunk. This means that computers can't store an infinite amount of information in a finite volume of space.	Mathematically a function is defined to be a set of pairs, so that, for example, the function square(n) = n^2 is defined to be the infinite set square = {(n, n^2) | n ε N} = {(0,0), (1,1), (2,4), (3,9),...}. Theoretical computer science rejects this definition as unworkable in practice, because it requires the construction and manipulation of infinite objects. Instead, a function is a rule or process by which a collection of arguments is manipulated in such a way as to evaluate a result. Sometimes the evaluation process fails to generate an answer, and in that case, we say the function is undefined (for that particular collection of arguments). The process by which functions are evaluated are called computer programs. Those (mathematical) functions that can be evaluated in the computer-science sense are called computable functions; those that can't are called

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			uncomputable. The time used for executing instructions is divided into discrete chunks whose length is always greater than some positive minimum length. Processing proceeds in discrete steps, one per chunk of time. This simply means that computers can't perform infinitely many instructions in finite time.
1.1.13. Representations can represent the infinite			The simplest representations stand for a single entity or a finite set of entities, e.g. a single number can be represented by a finite binary string and a set of experimental data by a series of binary strings separated by markers. An algorithm can also be represented by a finite binary string. A grammar of a language is a finite description of a potentially infinite set of sentences (strings) in a language. A grammar can help us analyse a given string to see if it is part of the language. A

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			grammar can also be used to generate the strings of the language in some systematic order. The grammar is finite, the language infinite. A syntax description of the Pascal language represents the set of all allowable Pascal programs. An algorithm is another example of a finite description of the infinite. In this case, the entities represented are the computations that the algorithm can generate. This is what makes programming so difficult. The program's designer needs to be able to show that every computation in the infinite set meets the input-output specifications of the algorithm. Fractal scene generation and programming
1.1.14. Representations are equivalent if they represent the same information	Lossless and lossy compression of bit maps Wav files and mp3	Lossless and lossy compression of bit maps Wav files and mp3	A folded DNA strand is not equivalent to an unfolded (linear) strand. Lossless and lossy compression of bit maps Wav files and mp3
1.1.15. Linear and non-linear representations	Main memory is linear - flat memory Strings of characters	Main memory is linear - flat memory Strings of characters	Most representations occurring in nature are nonlinear, e.g. folded DNA. DNA as a natural representation of the basic

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			information to generate the cells of an organism. RNA reads the DNA by finding and following sub-sequences, producing new amino acids as instructed by the DNA. DNA-RNA illustrate two basic principles of representation: 1. Information is embodied in the physical configuration of a carrier 2. The interpretation of information is assigned by the observer Because the RNA cannot "see" certain important protein-pairs in unfolded DNA, some information present in the folded form is absent in the unfolded form. The 3D shape of the carrier is important to the information that can be perceived. This consideration is not present when we think of purely linear representations such as bit sequences
1.1.16. Achieving equivalence of stages of development of a program: **Requirements** **Specifications**	Specification of inputs and output Specification of algorithm in program flowchart Execution of program flowchart, e.g. Raptor, Yenka	Specification of inputs and output Specification of algorithm in pseudocode or program flowchart	Specification of inputs and output Specification of algorithm in pseudocode, structured English or program flowchart

Principle	Key Stage 3	Key Stage 4	Key Stage 5

Source language **Compiled code** **Compiled code = original requirements**			
1.2 REPRESENTATIONS CAN BE COMPRESSED			
1.2.1. **If a long representation L and a short representation S hold the same information, an algorithm for constructing S from L is called compression. If the process is reversible, the algorithm for recovering L from S is called decompression. If decompression fully and completely reconstructs L from S, the compression is lossless.**	Run-length encoding for text	Run-length encoding for text and images	Run-length encoding for text and images
1.2.2. One possibility for lossless compression is to substitute shorter codes for symbols encoded in L. For example, 8-bit ASCII codes in L might be replaced by variable length codes of average length 6 to construct shorter representations S. S can be decompressed to L simply by restoring the original 8-bit codes.	Morse coding	Huffman coding	Huffman coding

Principle	Key Stage 3	Key Stage 4	Key Stage 5
1.2.3. The shortest possible lossless code has average word length equal to the entropy of the symbols being encoded (Entropy = -p log p where p is the probability of a symbol).			
1.2.4. A second possibility for lossless compression is for S to be an algorithm which when executed generates L. Pseudo-random numbers are an example of algorithmic compression: L is the longest sequence of random numbers and S is the compact algorithm for generating them. Although the sequence satisfies statistical test for randomness, it cannot be truly random because S is so much shorter and orderly than L. Another example is the Mandelbrot set: an incredibly complex set of points generated by a very short algorithm.	Programming language library routine for generating random numbers	Programming language library routine for generating random numbers	Algorithm for generating a pseudo-random number Programming language library routine for generating random numbers Generating MandelBrot set using $z = z^2 + c$
1.2.5. Compression that is value-preserving but lossy: the compression eliminates low-value symbols from L. This	Human ear MP3 encoding and effect on file size and quality - suppresses frequencies that ear	JPEG image encoding and effect on file size and quality MP3 encoding and effect on file size and quality	JPEG image encoding MP3 encoding MPEG3 video encoding

Principle	Key Stage 3	Key Stage 4	Key Stage 5
requires valuation of symbols encoded in L and a value threshold to decide which symbols to preserve. In this case S may be considerably shorter than the entropy lower bound for L, but now L may not be accurately reconstructed from S.	cannot hear giving a compression ratio of ten or more MPEG3 video encoding and effect on file size and quality JPEG image encoding and effect on file size and quality	- suppresses frequencies that ear cannot hear giving a compression ratio of ten or more MPEG3 video encoding	
1.2.6. A value-preserving compression may suppress detail without reducing complexity. Complexity concerns interactions among components; detail the minutiae of components. For example, a model of a software system that displays only the module functions and their interactions will suppress many details, but will not reduce the complexity of the software system - the modules still exist.		Modular development of systems - Hiding complexity behind an interface	Modular development of systems Procedure interfaces - Hiding complexity behind an interface Objects and their interfaces
1.3 FINITE REPRESENTATIONS OF REAL PROCESSES ALWAYS CONTAIN ERRORS			
1.3.1. A representation of a continuous real variable or process can never be exact because the finite number of		Unsigned fixed point binary representation of numbers with a fractional part	Floating point numbers are a basic way of representing numerical data in a machine. A floating point number consists of a mantissa M

Principle	Key Stage 3	Key Stage 4	Key Stage 5
represented points cannot cover the infinity of real points. **All fixed point and floating point arithmetic therefore induces representation errors between the numbers in the machine and the numbers in the real process.** **Representation error is the difference between a real number and its nearest represented number.** **With careful planning, algorithms can be organised so that representation errors of outputs are limited.** **Mathematical software libraries are designed this way.**		Representation error Issue with using a number with a fractional part as a loop control variable Range of fixed point numbers in a given number of bits	between 0 and 1 and an exponent E, meaning M x 2^E. Suppose that E and M are stored as parts of 32-bit words. Then there are at most 2^{32} floating point numbers. Therefore the error between a real number and the nearest floating point number representing it has an error of as much as 2^{-32}. For N-bit words, every floating point has an error of as much as 2^{-N}. Significant digits Precision Special cases, e.g. representing zero Rounding errors Cancellation errors Underflow Overflow IEEE standard for floating-point numbers Normalisation Reasons for normalisation Comparison of fixed and floating point ranges Mathematical software libraries

Principle	Key Stage 3	Key Stage 4	Key Stage 5
1.3.2. In a poorly organised long computation, representation errors can accumulate, culminating in a floating point result with a large error.			For example, a sum of n floating point numbers can have an error as high as n x 2^{-32}. Errors in small differences can also cause problems. For example, an algorithm that computes 1/(A - B) may give a divide-by-zero error if A and B are within 2^{-32} of each other.
1.4 INFORMATION RETRIEVAL			
1.4.1. Primary task of an IR system: **Retrieve documents with content that is relevant to a user's information need** **• Document set is fixed (size can vary from 10s of documents to billions)** **• Information need is not fixed (ad-hoc retrieval)**	Natural Language search - www.powerset.com	Natural language search - www.powerset.com	Meaning of IR - Indexing, retrieving and organizing text by probabilistic or statistical techniques that reflect semantics without actually understanding (James Allan, U.Mass) Models of IR – Boolean – Vector space – probabilistic Document Retrieval (ad-hoc retrieval) • Document Filtering or Routing • Document Categorisation • Document Summarising • Information Extraction • Question Answering

Principle	Key Stage 3	Key Stage 4	Key Stage 5

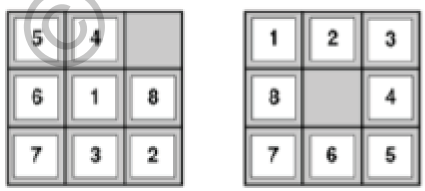
Learning outcomes for Key Stages 3 to 5 Computing. Strand: COMPUTATION

Learners should know and understand……

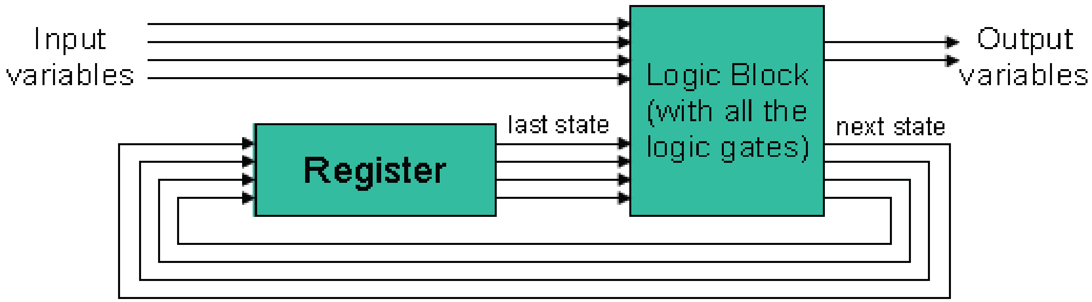
2. COMPUTATION PRINCIPLES			
2.1 COMPUTATIONS CAN BE OPEN OR CLOSED			
2.1.1. A computation is closed if its objective is to reach an end state in a finite time after being started in a beginning state.			Turing machine, input supplied on tape.
2.1.2. Closed computations are associated with mathematical functions that map beginning states to ending states.	Mathematical world view of computing- Algorithms	Mathematical world view of computing- Algorithms	Mathematical world view of computing- Algorithms
2.1.3. A computation is open if its objective is to continue indefinitely. In certain states, the computation can exchange information with the environment. Incoming requests from the environment, called tasks, alter the state of the computation; responses to	Interactive world view of computing- Event-driven programming in a gaming context such as Scratch, Gamemaker or Alice	Interactive world view of computing- Event-driven programming in for example - Python - Javascript - Delphi	Interactive world view of computing- Event-driven programming in for example - Python - Delphi - C#

Principle	Key Stage 3	Key Stage 4	Key Stage 5
tasks are output to the environment.		- Greenfoot - Blackfoot - VB.Net - StarLogo TNG Types of event and event handlers GUI programming - Widgets □ buttons □ labels □ checkboxes □ radio buttons □ data-entry field/textbox □ list box □ list box item □ comboboxes □ menu bar □ menu bar icon □ drop-down menu □ drop-down menu item □ Memo boxes □ labels	Events and the event queue -Client-server architecture - Object-oriented event-driven programming - GUI programming ▪ Registered handlers pattern ▪ Callback programming Maintaining state - stateless event driven application programming ▪ simple web page retrieval - stateful event driven application programming and state machines ▪ Web applications - e.g. shopping cart application ▪ GUI - e.g. checkbox ▪ Parsing a programming language or a mark up language - e.g. next character not in correct context - Ways to remember state ▪ State needs to be maintained only while application running - use in memory variables ▪ State needs to be maintained when execution suspended

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			□ use persistent medium, e.g. file or database □ delegate responsibility for remembering to caller. receive its state information from its caller when it starts and returns state information to its caller when it terminates □ e.g. Web application - create a session object and session ID on server, store session object in a database, send session ID to client in every web page.
2.1.4. Open computations are associated with interactive processes as well as non-terminating service processes such as web servers.	Use of a web server - starting and stopping a web server on a local machine, e.g. Apache web server - uploading web pages to a local and remote Apache web server - accessing uploaded web pages via a browser	Use of a web server - starting and stopping a web server on a local machine, e.g. Apache web server - uploading web pages to a local and remote Apache web server - accessing uploaded web pages via a browser	Use of a web server - starting and stopping a web server on a local machine, e.g. Apache web server - uploading web pages to a local and remote Apache web server - accessing uploaded web pages via a browser Creating and running a TCP server, e.g. in Delphi

Principle	Key Stage 3	Key Stage 4	Key Stage 5
2.2 COMPUTATION IS A SEQUENCE OF REPRESENTATIONS			
2.2.1. Computations are not just manmade products of manmade computers. **Computing machines made by man are just one way of realising computations**	Meaning of term algorithm Requirement for precision in expression of an algorithm Ways of expressing an algorithm: Flowchart Structured English Meaning of term computation Computation in nature	Meaning of term algorithm Independence of algorithms from any programming language Requirement for precision in expression of an algorithm Different ways of expressing an algorithm: Structured English Pseudocode Meaning of term computation Simple algorithms	Meaning of term computation Meaning of term algorithm Turing machine: finite control unit, an infinite tape, and a read-write head. Computation in nature
2.2.2. A computation is a sequence of states of a data representation caused by an algorithm. States represent values. **Successive representations are controlled by logic rules embodied in operators.** **An operator causes a specific, precise change in total state.**	Non-computer-based application of finite state automata and algorithms **The Eight Puzzle**  Start state Goal state	Finite state automata, e.g. ball-point pen, combination lock, lift control system, traffic lights What do these things have in common? – The state changes over time. – Need memory to capture the state of the system at any point in time. Finite state automata and algorithms in a digital computer context, e.g. traffic lights	The Finite State Machine combines a look-up table (constructed with binary logic) with a memory device (to store the state). • Components of the finite state machine: – Set of states: S = $\{s_1, s_2, \ldots, s_n\}$ – Set of observations: O = $\{o_1, o_2, \ldots, o_n\}$ – A transition function: describing how the state changes in response

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	Need to define sequences of moves to go from start configuration to goal configuration Need to represent any configuration in memory But need a way to describe how the configuration (i.e. state) changes over time		to the observation seen. Observations are set through inputs. Transition function represented by a transition table using logic gates. – Input = bits representing the observation and the last state. – Output = bits representing the next state. Register is used to store bits. It has an additional timing input that tells it when to change state (e.g. a clock that ‘ticks’ every second). The logic block implements the transition function. Finite State Machines with and without output,

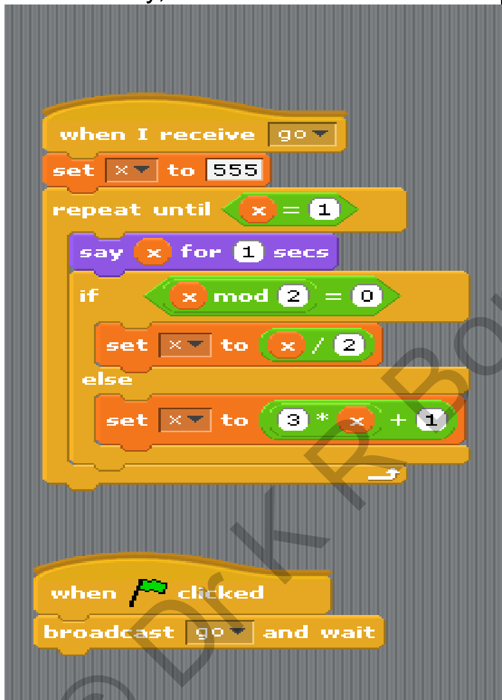


Principle	Key Stage 3	Key Stage 4	Key Stage 5
2.2.3. A computing machine is a physical system or process for holding a representation and acting upon it by an algorithm. **The "machine" can be based on digital electronics or it can be something else, e.g. DNA** **Machine instructions are machine-level operators**	Three-box model: Processor Storage - immediate access Input/Output electronics for communicating with other devices Use of a bus to interconnect the three boxes A DNA sequence is a representation and the various enzymes that split, read and recombine DNA strands are operators	Logic gates and simple combinational logic gate circuits - AND, NAND, OR, NOR, NOT, EOR - D-type flip flop Truth tables Three-box model: Processor Storage - immediate access I/O controllers for communicating with other devices Use of a bus to interconnect the three boxes System Clock Stored program computer Types of stored program computer: Harvard Princeton - von Neumann	Combinational logic circuits and De Morgan's laws, identities for Boolean variables Sequential logic - edge-triggered D-type flip flop Von Neumann architecture Processor and its internal architecture: Registers Control unit and instruction decoder Arithmetic and Logic Unit Internal bus Main memory Cache memory I/O controllers, I/O Ports and peripherals External bus or system bus: data bus, address bus, control bus Types of immediate access memory - RAM and ROM/EEPROM Secondary storage Stored program concept and Fetch-Execute cycle Types of stored program computer: Harvard

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			Princeton - von Neumann
2.2.4. An operator causes a specific, precise change in total state. **Machine instructions are machine-level operators. Most operators alter a confined, finite portion of a state. Some operators can alter the entire state** **Given an initial state, an algorithm specifies how operators from a finite set are applied to produce a final state: in what order and how many times.**	Very limited machine level instruction set for a very simple simulated machine: OUT, LOOP, ADD, SUBTRACT, SET, JUMP, HALT Simple machine code program to control lights for a simulated machine Simple assembly language program to control lights for a simulated machine Execution of simple machine code programs for a simulated machine Equivalence of assembly language program and machine code program	Limited instruction set for a simulated register machine: ADD, SUBTRACT, LOAD, STORE, OUT, IN, CMP, BEQ, BNE, JUMP, HALT Machine code program to control lights for a simulated register machine Assembly language program to control lights for a simulated register machine Execution of simple machine code programs for a simulated register machine Equivalence of assembly language program and machine code program	Instruction set for a simulated register machine with immediate, direct, indirect addressing and interrupts: ADD, SUBTRACT, LOAD, STORE, OUT, IN, CMP, BEQ, BNE, JUMP, JSR, RTS, HALT Machine code program to control lights for a simulated register machine Assembly language program to control lights for a simulated register machine Execution of machine code programs for a simulated register machine Equivalence of assembly language program and machine code program
2.2.5. An algorithm's total state is a record of the values of all its input, output, internal (private) variables and external variables. These variables are specified in the algorithm's data representation.	Very simple algorithms Specifying input Specifying output Concept of a variable Tracing a flowcharted algorithm,	Algorithms to illustrate tracing Specifying input Specifying output Simple roles of variables Hand-execution/Tracing an	More algorithms to illustrate tracing Specifying input Specifying output Roles of variables Hand-execution/Tracing an

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	watching change of variable values	algorithm	algorithm
2.2.6. Organise programs so that their dynamic computations mirror their textual structure in order to make algorithms more understandable and to reduce errors. **Control structures expressed with just four basic forms:** **Procedure call** **Sequence** **Selection** **Iteration**	Procedures Selection - If Then, If Then Else, Case/Switch Definite and indefinite iterations: For loops, While/Repeat loops	Procedures/Functions Selection - If Then, If Then Else, Case/Switch Definite and indefinite iterations: For loops, While/Repeat loops	Procedures/Functions Procedure/Function interfaces Selection - If Then, If Then Else, Case/Switch Definite and indefinite iterations: For loops, While/Repeat loops
2.2.7. Data-oriented computational forms **Computations driven by interactions with the external environment of a computation and by particular data that the computation receives at each interaction point**	Simple SQL queries on a data set: Select From Where >, >=, <, <=, =, <> World Wide Web and search engine searches Interactive multiplayer games	SQL queries on a data set: Select From more than one table Where with multiple conditions And/Or Nested queries using In DDL: Create database Create table Interactive multiplayer games	SQL queries on a data set: Select From more than one table Where with multiple conditions And/Or Order By, Group By Nested queries using In Insert, Update and Delete Aggregate functions Create database Create table Drop table

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			Drop database Create index DNA transcription
2.2.8. If an algorithm's data representation has an infinite number of states, the algorithm can generate a potentially infinite number of computations. Thus an algorithm is a highly compressed representation of a very large, potentially infinite, space of computations. This is why algorithms are difficult to understand and prove correct	Look up tables for simple problems with finite set of inputs	Look-up tables Computable and non-computable problems Simple decision problems -decidable -undecidable	Finite problems are solvable: - Any algorithmic problem with a finite set of inputs is solvable - A table mapping each of the n inputs to the appropriate output is all that is required Algorithmic problems that have infinite sets of legal inputs are less accommodating - non-computable problems, e.g. tiling problems - computable problems Decision problems - decidable - undecidable

Principle	Key Stage 3	Key Stage 4	Key Stage 5
2.2.9. Computation is unavoidable: **The only general method of approaching the question of whether computations halt or produce useful results is to run them and see what happens. This conclusion gave birth to Computer Science.**	But currently, no one knows if this  program always stops!	An algorithm that may or may not go into an infinite loop: 1. Input x { x > 0} 2. While x is not equal to 1 Do If x is even Then divide x by 2 (integer division) Else set x to 3x + 1 x = 9 terminates: x1=9, x2=28, x3=14, x4=7, x5=22, x6=11, x7=34, x8=17, x9=52, x10=26, x11=13, x12=40, x13=20, x14=10, x15=5, x16=16, x17=8, x18=4, x19=2, x20=1 x = 22 terminates?	Syracuse conjecture: For all n>0, Syracuse(n)>0 Syracuse(n) = least i such that s1=n, …, si=1, if it exists or = 0 if si ≠1 for all i. - Easy case: Syracuse(2k) = k+1 for any integer k ≥ 0 - But not so easy for numbers which are not powers of 2! Problem: – If there exists an n such that Syracuse(n) = 0, we might not be able to prove it. Halting problem: An algorithm is a piece of text. An algorithm can receive text as input. An algorithm can receive an algorithm as input. The Halting Problem: Given two texts A,B, consider A as an algorithm and B as an input. Will algorithm A halt (as opposed to loop forever) on input B? Theorem: No algorithm can decide

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			the Halting Problem. There are many problems that turn out to be undecidable. – All involve computations that might take an infinite number of operations to solve and you're never quite sure when to stop. It is useful to know which programs you should run, and which programs you shouldn't run! Showing that a problem is decidable often involves showing that this problem is analogous to another problem which we already know is decidable or not. E.g. Post Correspondence problem (PCP) is not decidable because it is analogous to the Halting Problem.
2.2.10. Turing's concept of a universal machine - a machine that can reproduce the closed computations of any other machine. The rules of operation and data of the subject machine are encoded as standard-form representations; the universal machine interprets and updates the representations, applying the logic rules of the subject machine	Scripting language	Interpreters Scripting languages, e.g. Javascript Program as data Java applet - download as data, execute as a program Viruses - download as data execute as a program	Turing's universal machine Program as data Viruses Scripting languages - e.g.Javascript Java applets - byte code DNA transcription

Principle	Key Stage 3	Key Stage 4	Key Stage 5
2.2.11. Church-Turing thesis: A consequence of the universal machine is that any computing machine can simulate any other and any closed computational process (i.e. computing of functions) can be simulated by a computing machine. Church-Turing thesis does not apply to arbitrary procedures and processes which may be open, non-terminating and involve multiple inputs interleaved with outputs. Church lambda calculus, Post's production systems and Kleene's recursive functions are all equivalent		Limits of computation	The numerical functions that can be evaluated by "human clerical labour, working to a fixed rule and without understanding" are precisely those functions that can be evaluated by computer (i.e. universal Turing machine) All other types of computing machine are reducible to an equivalent Turing machine No physical computing device can be more powerful than a Turing machine. If a Turing machine cannot solve a yes/no problem, nor can any physical computing device. Functional programming and Lambda calculus - Scheme programming language
2.2.12. The interrupt mechanism gives computations an orderly way to interact with their external environments. Signals from the environment trigger a procedure call on an operating system routine that receives and acts properly on the incoming message. The interrupt is an elegant way to receive unpredictable data within a control-structure environment			Interrupts - maskable and non-maskable - interrupt priorities - identifying source of interrupt - vectored interrupts and vector table - disabling and enabling interrupts

Principle	Key Stage 3	Key Stage 4	Key Stage 5
2.3 Computations have characteristic speeds of resolution			
2.3.1. Resolution time for a closed computation is the time to reach an end state, after starting at a beginning state.	Computation has limits Resolution time depends upon size of data set and the algorithm used to process the data	Computation has limits Resolution times for different algorithms with differently sized data sets	Computation has limits Resolution times for different algorithms with differently sized data sets
2.3.2. Resolution time for a open computation means the time to respond to a task, after the task is submitted.	Downloading web pages containing images with different file sizes	Downloading web pages containing images with different file sizes	Downloading web pages containing images with different file sizes
2.3.3. Resolution times for closed computations are expressed with order notations that express the way the resolution time depends on the size of the input. Computations whose time-complexity order is polynomial ($O(n^k)$ or better) are classified as "tractable". Computations whose order is exponential ($O(2^n)$ or worse) are classified as "intractable".	Non-mathematical comparison of polynomial algorithm with an exponential algorithm Use of an exponent calculator	Non-mathematical comparison of polynomial algorithms with exponential algorithms - Linear search - Towers of Hanoi	Tractable problems Intractable problems - using approximate solutions Polynomial time-complexity problem Exponential time-complexity problem NP problems
2.3.4. Some mathematical functions are non-computable - they cannot be computed at all. They have no resolution time.	Logic puzzles which are non-computable, e.g. the Barber paradox	Finding an odd perfect number	Halting problem Tiling problem

Principle	**Key Stage 3**	**Key Stage 4**	**Key Stage 5**
Examples include program halting, program equivalence, maximum possible running time and finding Kolmogorov complexity.			
2.3.5. Because they are not intended to stop, open computations use different measures of speed. The two most common are response time and throughput. Response time measures from when a user submits a task to a process and receives an answer. Throughput measures the number of tasks per unit time that the process completes.	Downloading web pages containing images with different file sizes	Interactive operating systems Network OS Batch processing	Multiprogramming operating systems Multitasking operating systems Thin client multiuser operating systems Interactive operating systems Network OS Batch processing
2.4 COMPLEXITY MEASURES THE TIME OR SPACE ESSENTIAL TO COMPLETE COMPUTATIONS			
2.4.1. Complexity measures the time and space required to complete an algorithm's computations.	Summing natural numbers by addition and by formula.	Finding factors by hand	Finding factors programmatically Timing an execution
2.4.2. Standard measure of algorithm complexity applies not to the algorithm but to its computational space.	Reversing elements of a list by hand Sorting elements of a list by hand	Reversing elements of a list programmatically Sorting elements of a list	Units for measuring time = basic operation Order of growth

Principle	Key Stage 3	Key Stage 4	Key Stage 5
Complexity expresses the relationship between execution time (or space) and the size of the input. The relation is expressed as "on the order of" - denoted O(.) - a function giving worst-case time (or space) for an algorithm to achieve its result.		programmatically Timing an execution	Asymptotic behaviour Bubble sort algorithm takes time proportional to n(n-1)/2 to arrange a list of n items into ascending order and can do this in the same memory as its input. Therefore time complexity of bubble sort is $O(n^2)$ and the space complexity is O(n).
2.4.3. An example of a simple algorithm with complex behaviour is the request to print all n-digit numbers. The algorithm is short. Its execution time is $O(10^n)$ because it enumerates all the numbers. The algorithm also needs space $O(10^n)$ to contain the answer. Numbers raised to powers grow so rapidly, therefore, a small request can create impossible demands.	Complex behaviour of a simple algorithm: Generate all 3-digit numbers, then all 4-digit numbers and so on. Then use PERMUT function in Excel spreadsheet to check answers.	Complex behaviour of a simple algorithm: Time to generate all 30-digit numbers is $O(10^{30})$ steps which on the fastest supercomputers (10^{12} operations per second) would take 10 billion years. For all 10-digit numbers, which could be computed within a second, the output would require 2000 reams of paper at 1000 numbers per page.	Complex behaviour of a simple algorithm: Programmatically generate all n-digit numbers Time and print.
2.4.4. We can lump together all algorithms of the same time (or space) complexity to form complexity classes:			Order of complexity Linear algorithms - order O(n) Quadratic algorithms - order $O(n^2)$ Polynomial algorithms - order $O(n^k)$ Exponential algorithms - order $O(2^n)$

Principle	Key Stage 3	Key Stage 4	Key Stage 5
2.4.5. Algorithms of complexity O(2^n) or worse are considered "intractable". Algorithms of complexity O(n^k) or better are considered "tractable". This division of hard versus easy is relative; even an O(n^2) algorithm can demand more time than we can give when n is large enough.		Qualitative understanding of complexity of algorithms	Classification of problems into tractable and intractable Why O(2^n) problem is considered intractable and O(n^k) is not given size of input n and under what circumstances
2.4.6. Some intractable algorithms are slow because they have to enumerate many cases in search of a small set of optimal cases. These algorithms contain "choice points" at which they can select one of several possible configurations for testing. An algorithm has to remember its choices so that, when one does not work out, it can backtrack and try a different choice. The algorithm could avoid backtracking if it could successfully guess the best choice the first time it encounters a choice point. Such an algorithm is called nondeterministic. Numerous intractable deterministic			Backtracking algorithms Depth-first search Breadth-first search Alpha-beta pruning

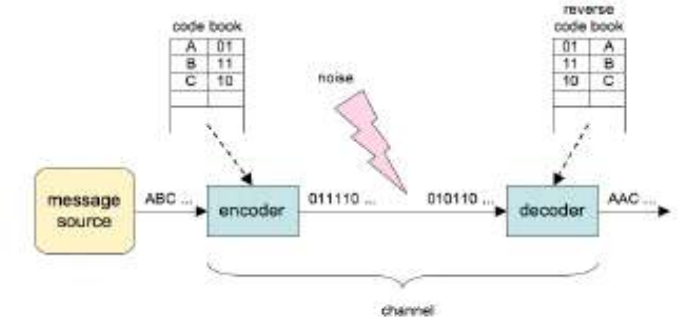
Principle	Key Stage 3	Key Stage 4	Key Stage 5
algorithms have tractable nondeterministic counterparts. For these algorithms, an answer can be verified in polynomial time even if it takes exponential time to compute.			
2.4.7. Tractability can be extended from individual algorithms to problems. A problem's complexity is the complexity of the best algorithm for solving it. P denotes the class of problems that are solvable with deterministic polynomial-time algorithms and NP the class of problems solvable with nondeterministic polynomial-time algorithms. It is unknown whether P = NP. At present, the best algorithms known for NP problems are exponential or worse.			P versus NP Guess a solution check guess in polynomial time
2.4.8. Heuristics are polynomial-time algorithms that employ simple rules of thumb to find approximate solutions to NP problems. While not guaranteed to find answers even close to optimal, heuristics are often good enough for			Solving scheduling and timetabling problems.

Principle	**Key Stage 3**	**Key Stage 4**	**Key Stage 5**
practice. Experimental studies reveal which heuristics work well in practice.			
2.4.9. Computation A is reducible to computation B if A can be solved by encoding it as an instance of B; the encoding process should take no worse than polynomial time.			Finding minimum number of colours to colour a map, finding minimum number of mobile phone frequencies, finding minimum number of fish aquaria all reduce to the same graph representation to which the same algorithm can be applied.
2.4.10. There is a subset of NP called NP-complete (NPC). NPC problems are considered the hardest of the NP problems because every NP problem can be reduced to at least one of them. Moreover all the NPC problems are mutually reducible to one another. Over 3000 problems belong to NPC, including typical engineering, science and commerce problems. Although no one knows of a fast (deterministic polynomial) algorithm for any member of NPC, if anyone ever finds one, it can be converted to a fast algorithm for every other member of NPC; that would			Some examples of NP-complete problems: Travelling Salesman problem

Principle	Key Stage 3	Key Stage 4	Key Stage 5
definitively prove that P = NP. That no fast algorithm has been found for any NPC problem is taken as strong empirical evidence that P ≠ NP			
Learning outcomes for Key Stages 3 to 5 Computing. Strand: COMMUNICATION and STORAGE ***Learners should know and understand……***			
3. COMMUNICATION and STORAGE PRINCIPLES			
3.1 INFORMATION CAN BE ENCODED INTO MESSAGES			
3.1.1. A message is a representation intended to communicate information	Messaging through the ages: Drum beats Smoke signals Naval signal flags Aldis lamp signalling Morse code Telegraph Telex Teletex/Oracle Email	Messaging through the ages HTML and CSS	Messaging through the ages HTML and CSS XML

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	SMS Mobile phone text messaging Skype Twitter HTML and CSS		
3.1.2. Messages are encoded into signals that move through a medium. Messages are abstract, signals concrete. Signals can be decomposed into profiles of continuous sine waves of various frequencies and amplitudes.	Types of transmission media Copper wire - twisted pair Coaxial cable Fibre optic Electromagnetic waves -very low frequencies to microwaves	Types of transmission media Copper wire - twisted pair Coaxial cable Fibre optic Electromagnetic waves -very low frequencies to microwaves	Fourier series. Time domain. Frequency domain. Discrete Fourier transform. Fast Fourier transform. Frequency domain for a square wave. (A Level Physics) Attenuation of signal. Bandwidth.
3.1.3. A digital sampling of a waveform records amplitude at periodic sampling times. Digital sampling loses no information if the sampling rate is at least twice the highest frequency in the waveform (Nyquist's theorem).	Analogue data. Digital data. Analogue signal. Digital signal. Analogue to digital converter. Digital to analogue converter. Digitisation of sound using a package such as Audacity or Goldwave	Analogue data. Digital data. Analogue signal. Digital signal. Analogue to digital converter. Digital to analogue converter. Digitisation of sound using a package such as Audacity or Goldwave	Analogue data. Digital data. Analogue signal. Digital signal. Analogue to Digital converter. Digital to Analogue converter. Sampling rate. Sampling resolution. Pulse Amplitude Modulation. Pulse Code Modulation. Advantages of PCM over PAM. Nyquist's theorem.

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.1.4. The human ear responds to frequencies up to 22KHz. Sampling sound at 44K (or more) samples per second preserves the audible frequencies. However frequencies beyond 22KHz also contribute to the reconstruction; losing them means some loss of fidelity in the reconstruction.	The human ear. The human ear is a natural MP3 encoder Investigation of different sampling rates using a package such as Audacity or Goldwave	Investigation of different sampling rates using a package such as Audacity or Goldwave	Investigation of different sampling rates using a package such as Audacity or Goldwave
3.1.5. Computer systems and networks use digital encodings. This does not constrain the capabilities of those systems because human-generated messages are composed from discrete symbols and because the brain (which perceives information) already samples its sensory input.	Almost everything can be modelled digitally or expressed in yes/no statements. Binary choices are very natural to realise in the physical world - light switches, punched cards, the pressing of a key to make Morse code, ticking a box in multiple choice tests and making a machine behave in a way where its behaviour patterns are controlled by simple choices is natural. All data can be modelled digitally(?), all processes can be modelled digitally(?).	Almost everything can be modelled digitally or expressed in yes/no statements. Binary choices are very natural to realise in the physical world - light switches, punched cards, the pressing of a key to make Morse code, ticking a box in multiple choice tests and making a machine behave in a way where its behaviour patterns are controlled by simple choices is natural. All data can be modelled digitally(?), all processes can be modelled digitally(?).	Almost everything can be modelled digitally or expressed in yes/no statements. Binary choices are very natural to realise in the physical world - light switches, punched cards, the pressing of a key to make Morse code, ticking a box in multiple choice tests and making a machine behave in a way where its behaviour patterns are controlled by simple choices is natural. All data can be modelled digitally(?), all processes can be modelled digitally(?).

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.2 DATA COMMUNICATION ALWAYS TAKES PLACE IN A SYSTEM CONSISTING OF A MESSAGE SOURCE, AN ENCODER, A CHANNEL, AND A DECODER			
			
3.2.1. See above	International signal codes -Morse codes -Baudot codes -Naval flag codes		
3.2.2. A channel is a communication medium that uses specific kinds of signals to represent information.	Serial communication -serial port -USB port Parallel communication	Serial communication -Start and stop bits, asynchronous communication -RS232/V24, mark and space	Handshaking protocol for -serial communication -parallel communication Ethernet - CSMD

Principle	Key Stage 3	Key Stage 4	Key Stage 5
		-RS423 -USB -Ethernet Parallel communication. Advantages and disadvantages	Synchronous transmission Time division multiplexing Frequency division multiplexing Manchester encoding
3.2.3. Noise represents any conditions that can disrupt signals, preventing their accurate reception			
3.2.4. A message source is a set of possible messages and their probabilities			
3.2.5. An encoder is a device that represents a message with a channel code, that is with signals in the channel. An encoder uses a "codebook" that associates a channel code with each message.	Serial port USB port Parallel port Computer keyboard and electronics converts key presses to a - 7-bit digital code, ASCII - 8-bit code - extended ASCII	Computer keyboard and electronics generate a scan code that is converted to - ASCII - Unicode	Codecs - GSM mobile phone speech codec - audio codecs mp3 - video codecs mp4 Gray code
3.2.6. A channel code is uniquely decipherable if and only if it has the prefix property:			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
no code is prefix of another. Decoders need uniquely decipherable channel codes for reliable reception.			
3.2.7. Tree codes, which assign messages to the leaves of binary trees, are prefix codes. The Huffman Code is a tree code that minimises average code length by assigning high-probability messages shortest path lengths.	Morse code		Huffman coding
3.3 MESSAGES CORRUPTED DURING TRANSMISSION CAN BE RECOVERED			
3.3.1. Noise on the channel can reverse or garble some of the bits. A changed or corrupted bit is called an error.	Causes of error in bit streams	Causes of error in bit streams	Causes of error in bit streams
3.3.2. An error correcting code contains extra, check, bits that reveal the original value of an erroneous bit.	Parity bit - Even - Odd	Parity bit - Even - Odd Parity checking in serial transmission	Parity bit - Even - Odd Parity checking in serial transmission Hamming code

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.4 MESSAGES CAN BE COMPRESSED			
3.4.1. Since messages are representations, they can be compressed and decompressed. In communication, compression means to re-encode a source file or bit stream with fewer bits.			
3.4.2. Lossless compression means every original bit is recoverable from a compressed file. Examples are Huffman and zip codes.	Simple techniques for compressing text	Lossless text compression techniques Run-length encoding for images	Lossless text compression techniques Lossless image compression
3.4.3. Lossy compression means a compression code for which some source bits are irrecoverable. Lossy compression algorithms attempt to delete information believed to be of no value to the receiver, so that the receiver will not notice the loss. Examples are JPEG(images) and MP3(sound)	Simple understanding of MP3 coding Working with MP3 sound files of different compression ratios and effect on file sizes and quality of sound Working with JPEG images of different compression ratios effect on file sizes and quality of image	Basic understanding of MP3 coding Basic understanding of JPEG encoding Recording and compressing sound files Effect on file sizes and quality of sound Working with JPEG images of different compression ratios and quality of image	Understanding of MP3 coding Understanding of JPEG encoding Working with MP3 sound files of different compression ratios and effect on quality of sound Working with JPEG images of different compression ratios and quality of sound

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.5 MESSAGES CAN BE ENCRYPTED			
3.5.1. Information in a code is hidden (enciphered) if the receiver has no fast algorithm for decoding. Therefore the receiver can only guess the message contained in the cyphertext, giving low probability of finding the hidden message.			Steganography - store a short message in a bit map image - incorporate bit map image in a web page
3.5.2. Encryption is a computation that converts messages into ciphers, under the control of one or more keys. Decryption by fast algorithm is possible if the receiver has the keys. Encryption is not used for channel encoding; that is a separate process.	Caesar cipher -algorithm -key Cipher wheel Sending messages using numbers Introduction to modular arithmetic Breaking Caesar ciphers using letter frequencies	Substitution ciphers Breaking substitution ciphers using letter frequencies Vignere ciphers Factors, prime numbers, composite numbers, prime factorisation, use of factor trees, common factor, greatest common factor, relatively prime Using common factors to crack Vigenere ciphers Modular arithmetic - reduce mod n - congruent numbers - modular inverses - shortcut for multiplying in modular arithmetic	Multiplicative cipher keys should be relatively prime to the size of the alphabet Finding prime numbers Sieve of Eratosthenes Counting primes and Euclid's theorem Twin primes Mersenne primes and GIMPS Sophie Germaine primes Large primes needed in the RSA cipher Raising numbers to powers in modular arithmetic

Principle	Key Stage 3	Key Stage 4	Key Stage 5
		Applications of modular arithmetic Multiplicative ciphers Using inverses to decrypt	
3.5.3. Secrecy means that no eavesdropper can decipher a message intercepted from the channel.			
3.5.4. Authentication means that the receiver can uniquely identify the sender of a message.			Use of public-private key encryption for authentication Message digest and digital signature Digital certificate Use of a salt to counter replay attacks
3.5.5. Single key encryption means a sender and receiver use the same key (symmetric encryption). Security depends on making key exchange secret. If a new key is used for every sender-receiver communication, this system provides both secrecy and authentication.		Symmetric encryption	Symmetric encryption
3.5.6. Public key encryption means the sender enciphers under one key and the receiver			Public-private key encryption RSA cipher

Principle	Key Stage 3	Key Stage 4	Key Stage 5
deciphers under a different key. The two keys, called public and private, are linked but knowledge of one does not enable computation of the other. Enciphering with the public key ensures secrecy (only the owner of the secret key can decipher). Enciphering with the secret key ensures authentication (anyone can verify who sent the message).			
3.5.7. Bit rates of single key channels are orders of magnitude faster than for public key channels. Therefore, most encryption systems use public key systems for distributing session keys to the parties, who then use high-speed single key channels for their communications.			Session keys
3.6 HIERARCHICAL NAMING SYSTEMS ALLOW LOCAL AUTHORITIES TO ASSIGN NAMES THAT ARE GLOBALLY UNIQUE IN VERY LARGE NAME SPACES			
3.6.1. Users have to deal with tens of thousands of file names in their own systems and	Postal addresses, phone numbers and organisational charts illustrate non-computational hierarchical naming	Postal addresses, phone numbers and organisational charts illustrate non-computational hierarchical	Namespaces

Principle	Key Stage 3	Key Stage 4	Key Stage 5
(potentially) billions or trillions of names in the full Internet. The hierarchical naming principle is a way of constructing a local name that is unique within the entire name space.	systems. World telephone system is hierarchical. - country codes - national numbering plan □ area numbering □ local numbering Postal codes	naming systems. World telephone system is hierarchical. - country codes - national numbering plan □ area numbering □ local numbering Postal codes	
3.6.2. The hierarchical naming principle organises all objects in the name space into a tree whose internal nodes are directories. A directory is a list of object names and their handles, with no duplicate names. Pathnames in such a hierarchy are unique.	Computer directory/folder structure. Pathnames. In computers the directory structure is hierarchical. The system administrator (root) can create new user names and give each user a directory. Each user can choose file names and place them in the directory. The user can also define subdirectories and act as the authority that chooses names within these. In the resulting tree, directories are internal nodes and files are leaves. A pathname is the sequence of labels from the root to the named directory or file.	Computer directory/folder structure. Pathnames. The Internet host-naming system is also hierarchical. An international authority (ICANN, the International Corporation for Assigned Names and Numbers) determines the top-level domains and their authorities. There are seven generic top-level domains (.com, .edu, .gov, .int, .mil, .net, .org) and several more specialised top-level domains. Country domains Domain registrars.	Computer directory/folder structure. Pathnames. The Internet host-naming system is also hierarchical. An international authority (ICANN, the International Corporation for Assigned Names and Numbers) determines the top-level domains and their authorities. There are seven generic top-level domains (.com, .edu, .gov, .int, .mil, .net, .org) and several more specialised top-level domains. Country domains. Domain names. Fully qualified domain names. Domain registrars.

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.6.3. A directory hierarchy maps pathnames to handles			Pathnames → Handles Handle - Every object in system given a unique, permanent, never-reused name. Local machine on which a file is created already has a unique numerical hardware address. Concatenate this with a time-stamp from the local computer's clock. Since at most one file creation can occur in one clock tick, the resulting number will not be used for any other file. Handles stored in directories; users never see these and are not responsible for maintaining their integrity. NFS filing systems
3.6.4. These pathnames are global symbolic names for objects. Every object has a unique pathname. Therefore pathnames can be used to share objects.			Namespaces
3.6.5. The Internet is a large name space with URLs as the names. A URL(Uniform Resource Locator) is a hostname concatenated with a pathname	URLs	URLs, URIs, URNs URN - urn:isbn:978-0-7487-8296-3 (A URN can be used to talk about a resource without implying its location or how to access it)	URLs, URIs, URNs URN - urn:isbn:978-0-7487-8296-3 (A URN can be used to talk about a resource without implying its location or how to access it)

Principle	Key Stage 3	Key Stage 4	Key Stage 5
in the host's directory tree. This embeds all host trees as branches from a global Internet tree.		URL - ftp://ftp.test.com/RomeoAndJuliet.pdf (implies that a user can get a representation of that resource via FTP from a network host named ftp.test.com)	URL - ftp://ftp.test.com/RomeoAndJuliet.pdf (implies that a user can get a representation of that resource via FTP from a network host named ftp.test.com)
3.6.6. Because pathnames can be reused, the same name can designate different objects at different times. Therefore, the Internet URL naming system does not guarantee that a name one acquires points to the same object it did at the time of acquisition. For directory entries a version number added to a directory entry enables access to old versions of an object.	Reusability of pathnames issues Broken links	Reusability of pathnames issues Broken links	Reusability of pathnames issues Broken links
3.7 ACCESS TO STORED OBJECTS IS CONTROLLED BY DYNAMIC BINDINGS BETWEEN NAMES, HANDLES, ADDRESSES AND LOCATIONS			
3.7.1. An essential part of every storage system is a means of naming objects and assigning them addresses in the storage system. All operations in the storage system - reading,	Physical drive names, logical/virtual drive names, hostnames, filenames, directory/folder names domain names, fully qualified domain names		

Principle	Key Stage 3	Key Stage 4	Key Stage 5
writing, relocating in the hierarchy, searching - rely on naming and addressing.			
3.7.2. A storage object is a data container in some agreed format with a set of allowable operations. For example, a page is a block of consecutive addresses of an agreed fixed length; the allowable operations are read and write for offsets. A file is a sequence of bytes; the allowable operations are open, close, read, write. A directory is a set of entries that associate symbolic names chosen by users with internal addresses of storage objects; the allowable operations are enter, remove, rename and search.		RAM pages Disk blocks File = sequence of bytes	RAM pages Disk blocks File = sequence of bytes
3.7.3. A virtual (storage) object is a simulation of the object using the available mechanisms of the storage system. A virtual memory, for example, simulates a large main memory using a small RAM, a hard disk, an address mapper, a page fault handler interrupt routine and a page replacement algorithm.		Virtual memory technique	Virtual memory technique

Principle	**Key Stage 3**	**Key Stage 4**	**Key Stage 5**
The processor simply issues read or write requests for addresses in the virtual memory; the system automatically converts each request into an appropriate sequence of address mappings, UP commands and DOWN commands.			
3.7.4. Modern storage systems, from those on individual computers to the entire Internet, use several levels of abstraction (virtualisation) to realise all the goals of the system: **Names** **Handles** **Addresses** **Locations**	School-based analogy: Student possesses a name assigned at birth and registered on birth certificate - Fred Bloggs (name); a person is assigned a unique number at birth for healthcare reasons - NHS number (handle); student joins a school and is assigned to a class - e.g. 7R (address); the class is assigned to room 2 (location); after a year the class is redesignated 8R(new address) but the class remain in the same location - room 2. In the real world, names are chosen by someone to refer to a person; people live in buildings that authorities assign addresses to; the buildings are physically located in the world - latitude and longitude. Authorities may change the address assigned to a building; people may change their name by deed poll but a building normally has a fixed	Objects are assigned handles - e.g. in object-oriented systems, e.g. window handles, file handles	Objects are assigned handles - e.g. in object-oriented systems, e.g. window handles, file handles

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	location.		
3.7.5. Names: **Symbolic strings chosen by users to name their objects.** **Users have extreme difficulty remembering machine-readable addresses (strings of bits). This level lets them give their own names for objects; the storage system maps the names to locations and accesses the objects for the user.**	User-chosen symbolic names for objects. Names are important to users who want to choose identifying strings that mean something to them and are easy to remember. Filenames, directory/folder names Variable names in programs Class and object names	Filenames, directory/folder names Variable names in programs Class and object names in programs Subroutine names Data type names in programs	Filenames, directory/folder names Variable names in programs Subroutine names Data type names in programs Constant names in programs Class and object names in OOP programs
3.7.6. Handles: **System-generated identifiers that are globally unique and are never reused. Handles distinguish all objects and all versions of the same object, allowing anyone at any time to access the object regardless of any local names assigned to it.**	ISBN number NHS number; National Insurance Number; Unique Learner Number; Driving Licence Number Handle generating programs GUID/UUID	Handles are system-chosen, globally unique identifiers for objects. Handles are important in the sharing of objects, user-chosen names are not reliable for this purpose because users may use the same name for different objects. One way to form a handle is to join a time stamp with the identifier of the machine creating an object. Every object receives a unique, unchangeable global name called a handle. New versions of an object receive new handles. Handles are essential if sharing of objects is required. Handles ensure that	Handles are system-chosen, globally unique identifiers for objects. Handles are important in the sharing of objects, user-chosen names are not reliable for this purpose because users may use the same name for different objects. One way to form a handle is to join a time stamp with the identifier of the machine creating an object. Every object receives a unique, unchangeable global name called a handle. New versions of an object receive new handles. Handles are essential if sharing of objects is required. Handles ensure that

Principle	Key Stage 3	Key Stage 4	Key Stage 5
		sharing is easy and care-free. Programming with file and window handles GUID/UUID	sharing is easy and care-free. Programming with file and window handles, GUID, UUID How does a file become part of an address space? One common method is to have an executing computation "open" the file. This causes the operating system to temporarily bind the computation to the file. The binding is erased when the computation "closes" the file.
3.7.7. Addresses: **Bit-strings identifying individual bytes of an address space. The compiler which creates the address space, initially generates them and thereafter the processor issues them as it executes instructions**	Addresses are bit-strings identifying individual bytes of an executable program. Addresses identify all the bytes in compiled programs and in the data used by those programs.	Addresses are bit-strings identifying individual bytes of an executable program. Addresses identify all the bytes in compiled programs and in the data used by those programs.	Addresses are bit-strings identifying individual bytes of an executable program. Addresses identify all the bytes in compiled programs and in the data used by those programs.
3.7.8. Locations: **Bit-strings used by the hardware to identify specific physical locations**	Locations are bit-strings identifying physical spaces on devices where data items are stored. Locations are physical sites for data. Physical memory addresses Range of physical memory addresses for a given processor system - width of address bus	Physical memory addresses Range of physical memory addresses for a given processor system - width of address bus	Physical memory addresses Range of physical memory addresses for a given processor system - width of address bus

<table>
<tr><th>Principle</th><th>Key Stage 3</th><th>Key Stage 4</th><th>Key Stage 5</th></tr>
<tr><td>3.7.9. Levels of virtualisation are represented as a series of dynamic maps:
• Name
• Handle
• Address
• location
The operations at each level are fixed but the mappings from the abstract state at one level to the representation at the next lower level change over time. Those mappings, called bindings, enable the same abstract state to be represented by many different configurations invisible to the user.</td><td><table>
<tr><th>Names</th><th>Handles</th><th>Addresses</th><th>Locations</th></tr>
<tr><td rowspan="4">Family Names</td><td>NHS number</td><td rowspan="2">Mobile phone number</td><td rowspan="4">Places</td></tr>
<tr><td>NI number</td></tr>
<tr><td>Driving licence number</td><td rowspan="2">Email address</td></tr>
<tr><td>Unique Learner number</td></tr>
</table></td><td>Windows task manager</td><td>Windows task manager</td></tr>
<tr><td>3.7.10. Mapping tables, which represent dynamic bindings, associate an item at one level with the corresponding item at the next lower level.</td><td>Just three tables required to handle mappings for the above diagram.
File directories/folders</td><td>Internet - Domain Name Service records the association between host-names and IP addresses
File storage: a file looks like a sequence of bytes. File broken into blocks that are stored on disk sectors. A file's blocks can be scattered across the disk. To find them, a File Access Table (FAT) is created that maps file blocks to disk sectors. A handle is created to identify the file and a table that maps handles to FATs. Finally, the handle</td><td>When a user creates a new object in a computer system, the system assigns the object a handle and binds it the name chosen by the user; the system also assigns a set of addresses to the object and binds them to the handle; and finally it binds these addresses to physical locations on a storage device. All these bindings can be changed as the system evolves. E.g. if an object is moved to a new position in the memory hierarchy, only the</td></tr>
</table>

Principle	Key Stage 3	Key Stage 4	Key Stage 5
		is stored in a directory along with the symbolic filename assigned by the user. The user can read or write the file by providing its symbolic name; the system uses the directory to map to the file's handle; the handle map to get the FAT and the FAT to get all the blocks of the file.	address-location binding needs to be updated. Virtual memory - page table records current associations between pages and memory frames. There are no handles. Compiler binds program names directly to addresses. Only the address-location binding is dynamic. In a file system, the directory structure records the associations between paths and handles.
3.7.11. In a dynamic map, an object named at a level may be unknown at that level. In this case the mapping tables yield a "not found" indicator when searched. Such a mapping fault creates an interrupt that invokes a process in the O.S. to find the missing object from the master copy in the lower levels of memory; thereafter, accesses to that object will map without fault.	Mobile phone moves from one cell to another during a phone call; base station must track movement so that phone call is not interrupted		In a virtual memory, every page table entry contains a presence bit set to 1 when the corresponding page is in RAM; if the page is missing, the bit is 0 and the page fault causes the O.S. to fetch the missing page from disk and update the page table.
3.7.12. Dynamic bindings provide location independence: neither a user nor a processor needs to know the physical location of an object to access	Location independence: Objects in the same system can be addressed the same way regardless of their physical location.	Non-contiguous allocation of disk blocks Defragmentation of a disk	Absolute load modules and absolute loader Relocatable code and relocating loader

Principle	Key Stage 3	Key Stage 4	Key Stage 5
it. Objects can be relocated to new locations by updating only the final map (address location)	Mobile phone location changes but phone can still be located. Can be used to track user. Mobile phone system allows owner to keep the same mobile phone number for life. SIM card which contains user's subscription information is simple transferred to new phone. IP addresses can be attached to different computers at different times.		Base register addressing Position independent code (relative addressing) Logical page numbers and offsets Dynamic relocation Non-contiguous allocation of RAM Non-contiguous allocation of file disk blocks Disk defragmentation Code sharing - dlls
3.7.13. In many cases, the bindings from names to addresses are static. A compiler changes all the symbolic variable names of a program to addresses. Symbolic names outside the module being compiled are labelled unresolved in an external symbol table; a separate linker (or make-file) program takes a set of separately compiled modules, resolves their external references by substituting addresses in other modules containing referenced objects and produces a complete address space ready to execute.	Mobile phone's IMEI number uniquely identifies a mobile phone and is permanently assigned to phone.	Separately compiled modules. Linkers. Dynamically linked libraries.	Separately compiled modules. Linkers. Dynamically linked libraries.

Principle	Key Stage 3	Key Stage 4	Key Stage 5
In this case the only binding that can be managed dynamically by the system is address → location.			
3.7.14. When viewed as a large storage system, the Internet conforms to the pattern • **URL/URI(name)** • **IP(address)** • **MAC(location)**	Names are URLs (uniform resource locators) or URIs(uniform resource identifiers), consisting of a hostname followed by the pathname of that object on the host. Addresses are 32-bit IP addresses, represented as four integers (0-255) separated by dots, for example 192.168.1.1. Locations are MAC (media access control) addresses assigned to network connector cards, e.g. an Ethernet address is 48 bits represented as six two-letter hexadecimal codes such as 00:0A:95:C4:1F:74.	Names are URLs (uniform resource locators) or URIs(uniform resource identifiers), consisting of a hostname followed by the pathname of that object on the host. Addresses are 32-bit IP addresses, represented as four integers (0-255) separated by dots, for example 192.168.1.1. Locations are MAC (media access control) addresses assigned to network connector cards, e.g. an Ethernet address is 48 bits represented as six two-letter hexadecimal codes such as 00:0A:95:C4:1F:74.	Names are URLs (uniform resource locators) or URIs(uniform resource identifiers), consisting of a hostname followed by the pathname of that object on the host. Addresses are 32-bit IP addresses, represented as four integers (0-255) separated by dots, for example 192.168.1.1. Locations are MAC (media access control) addresses assigned to network connector cards, e.g. an Ethernet address is 48 bits represented as six two-letter hexadecimal codes such as 00:0A:95:C4:1F:74.
3.8 DATA CAN BE RETRIEVED BY NAME OR BY CONTENT			
3.8.1. In addition to name-based addressing, memory systems also provide content retrieval for objects	We put data objects into storage systems because we want to retrieve them later. Telephone directory organised in ascending alphabetical name order. Telephone directory organised in	We put data objects into storage systems because we want to retrieve them later. Inverted lists for information retrieval from text documents - keyword table, constructed by scanning all the text documents, listing the	We put data objects into storage systems because we want to retrieve them later.

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	ascending telephone number order. Searching by name and by telephone number comparison of each of the above. Reverse lookup.	filename, page and line numbers at which each keyword appears. Someone could retrieve documents on particular subjects by giving all the keywords of interest.	
3.8.2. With content retrieval, the user specifies keywords or attributes and the memory returns a set of matching objects.		Database search - Moderately simple SQL queries through a query tool such as CuteSQL Internet searches	Database search - Advanced SQL queries through a query tool such as CuteSQL Internet searches
3.8.3. Data can be organised to facilitate fast retrieval	Use of an index An index, for example, maps keywords to lists of objects containing them A fast search of an index yields the matching object list quickly Binary search	Use of an index, e.g. Napster and Google An index, for example, maps keywords to lists of objects containing them A fast search of an index yields the matching object list quickly Binary search programmatically Hash table	Use of an index, e.g. Napster and Google An index, for example, maps keywords to lists of objects containing them A fast search of an index yields the matching object list quickly Binary search and binary search tree programmatically Hash table programmatically
3.8.4. Internet search is exceptionally demanding because search engines must infer broader meanings from a few keywords a user provides	Search engines for the retrieval mechanism Search engine quickly locates the URLs and opening sentences of documents containing keywords specified in search	Search engines for the retrieval mechanism Search engine quickly locates the URLs and opening sentences of documents containing keywords	Search engines for the retrieval mechanism Search engine quickly locates the URLs and opening sentences of documents containing keywords

Principle	Key Stage 3	Key Stage 4	Key Stage 5
and because the search database requires huge computing facilities to respond rapidly to queries.	criteria How many web pages are known to a particular search engine? Web crawlers Master keyword index.	specified in search criteria How many web pages are known to a particular search engine? Web crawlers Master keyword index.	specified in search criteria How many web pages are known to a particular search engine? Web crawlers Master keyword index. Internet search is much more challenging than information retrieval.
3.9 THE PRINCIPLE OF LOCALITY DYNAMICALLY IDENTIFIES THE MOST USEFUL DATA, WHICH CAN THEN BE CACHED AT THE TOP OF THE HIERARCHY			
3.9.1. Locality is the principle that a computation clusters its references during every phase of execution into small subsets of its objects, called locality sets.	A student working on a school project visits a library and locates several books in the stacks and brings them to the reading room. The student switches attention from one book to another as information is sought. Occasionally the student returns books to the shelves and additional books are obtained from the shelves. The student does a lot of studying from the books in the reading room in between visits to the shelves. Most of the student's attention is focussed on the books in the reading room. The reader's behaviour is called "locality". The student has localised attention on a small subset of the library, maintaining that attention on that subset	A student working on a school project visits a library and locates several books in the stacks and brings them to the reading room. The student switches attention from one book to another as information is sought. Occasionally the student returns books to the shelves and additional books are obtained from the shelves. The student does a lot of studying from the books in the reading room in between visits to the shelves. Most of the student's attention is focussed on the books in the reading room. The reader's behaviour is called "locality". The student has localised attention on a	A student working on a school project visits a library and locates several books in the stacks and brings them to the reading room. The student switches attention from one book to another as information is sought. Occasionally the student returns books to the shelves and additional books are obtained from the shelves. The student does a lot of studying from the books in the reading room in between visits to the shelves. Most of the student's attention is focussed on the books in the reading room. The reader's behaviour is called "locality". The student has localised attention on a

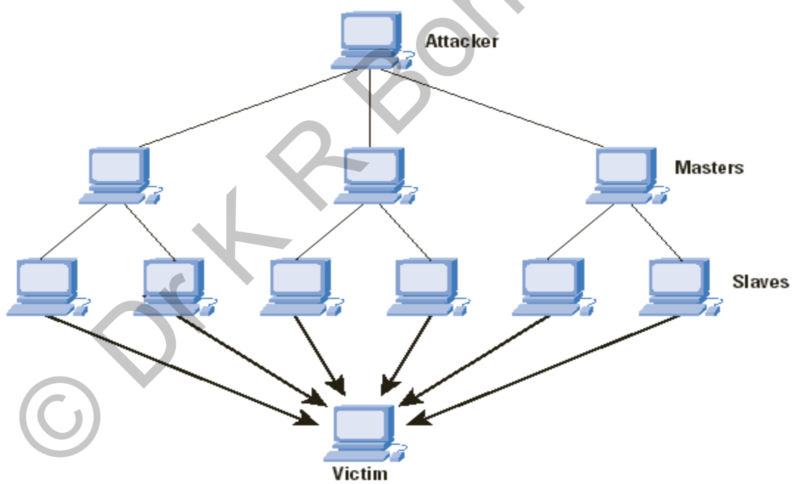
Principle	Key Stage 3	Key Stage 4	Key Stage 5
	for an extended period of time. Locality means that a person or a computation directs all its memory accesses to a small subset of objects for extended periods. **Locality is what makes the reading room effective.** Reading a book also illustrates the locality pattern. The reader limits attention to the pages of the book. While reading the book, the reader normally does not make random references to pages of other books. **Temporal locality** - maintain focus on task in hand **Spatial locality** - keep all the required objects close by Computations which act on behalf of human users, exhibit temporal and spatial locality. The storage system can take advantage of temporal locality by placing the most recently accessed objects at the top of the hierarchy for quick access. It can take advantage of spatial locality by storing related data in the same blocks of secondary memory.	small subset of the library, maintaining that attention on that subset for an extended period of time. Locality means that a person or a computation directs all its memory accesses to a small subset of objects for extended periods. **Locality is what makes the reading room effective.** Reading a book also illustrates the locality pattern. The reader limits attention to the pages of the book. While reading the book, the reader normally does not make random references to pages of other books. **Temporal locality** - maintain focus on task in hand **Spatial locality** - keep all the required objects close by Computations which act on behalf of human users, exhibit temporal and spatial locality. The storage system can take advantage of temporal locality by placing the most recently accessed objects at the top of the hierarchy for quick access. It can take advantage of spatial locality by storing related data in the same blocks of secondary memory.	small subset of the library, maintaining that attention on that subset for an extended period of time. Locality means that a person or a computation directs all its memory accesses to a small subset of objects for extended periods. **Locality is what makes the reading room effective.** Reading a book also illustrates the locality pattern. The reader limits attention to the pages of the book. While reading the book, the reader normally does not make random references to pages of other books. **Temporal locality** - maintain focus on task in hand **Spatial locality** - keep all the required objects close by Computations which act on behalf of human users, exhibit temporal and spatial locality. The storage system can take advantage of temporal locality by placing the most recently accessed objects at the top of the hierarchy for quick access. It can take advantage of spatial locality by storing related data in the same blocks of secondary memory.

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.9.2. Locality is a fundamental behavioural property of all computations. Its primary cause is limitation of human attention span; humans tend to approach problems by focusing on parts and solving them before moving to other parts. This is called temporal locality	Use of subroutines/subsystems.	Use of subroutines/subsystems.	Use of subroutines/subsystems.
3.9.3. The second cause of locality is organisation of data. Each data object is linked to a limited number of "neighbour" objects. A computation is most likely to access a neighbour of a current object in the near future. This is called spatial locality.	Use of a block-structured programming language and avoidance of use of GoTo. Every block should have one entry point and one exit point. Enforced in programming languages such as Scratch and StarLogo TNG which uses blocks that fit together like the pieces of a jigsaw.	Use of a block-structured programming language and avoidance of use of GoTo. Every block should have one entry point and one exit point.	Use of a block-structured programming language and avoidance of use of GoTo. Every block should have one entry point and one exit point.
3.9.4. A cache is a (hardware or software) memory device designed to hold locality sets of a computation. CACHE is very fast relative to the RAM.	The caching principle is used in nearly every computing technology. Computers put cache memory next to the processor chip so that most of the time the processor will find the data it needs nearby and can proceed at full speed. Local networks cache recently-used web pages on a local server, by passing a long access time to a remote server for many web page lookups. Web browsers retain copies of web pages viewed most recently. Application programs retain copies of	The caching principle is used in nearly every computing technology. Computers put cache memory next to the processor chip so that most of the time the processor will find the data it needs nearby and can proceed at full speed. Local networks cache recently-used web pages on a local server, by passing a long access time to a remote server for many web page lookups. Web browsers retain copies of web	The caching principle is used in nearly every computing technology. Computers put cache memory next to the processor chip so that most of the time the processor will find the data it needs nearby and can proceed at full speed. Local networks cache recently-used web pages on a local server, by passing a long access time to a remote server for many web page lookups. Web browsers retain copies of web

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	the files most recently referenced. Email programs retain lists of the email addresses used most recently. Video display cards retain a copy of the video image, updating only the changed bits rather than regenerating the entire image.	pages viewed most recently. Application programs retain copies of web pages viewed most recently. Email programs retain lists of the email addresses used most recently. Video display cards retain a copy of the video image, updating only the changed bits rather than regenerating the entire image.	pages viewed most recently. Application programs retain copies of web pages viewed most recently. Email programs retain lists of the email addresses used most recently. Video display cards retain a copy of the video image, updating only the changed bits rather than regenerating the entire image.
3.9.5. Working set is a measure of locality set. It is usually determined from the usage bits of objects during a fixed time window into the recent past. High performance memory management seeks to hold working sets in cache.			The "working set" of a computation at a given time is the set of objects referenced (read or written) during a time window into the recent past. The storage system will be most efficient when it measures the working sets of computations and places them at the top of the hierarchy. In 1959, the designers of the ATLAS Operating System at the University of Manchester proposed to automate the process of moving blocks of data. Their memory architecture came to be called virtual memory. Block transfers delegated to O.S. Experience has shown that virtual memories that retain working sets in main memory perform the best.

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			Virtual memory technique Swap spaces.
3.9.6. Locality principle tells us that some objects will be used more than others. With multiple users that implies queuing and congestion at the most popular objects. Therefore, having flat memory (no hierarchy) with all objects available in one space with uniform access times does not overcome this problem. The congestion can be avoided by caching copies of popular objects at multiple points in the flat memory space. This will reduce queuing delay.			Web caching
3.10 THRASHING IS A SEVERE PERFORMANCE DEGRADATION CAUSED WHEN PARALLEL COMPUTATIONS OVERLOAD THE STORAGE SYSTEM			
3.10.1. Thrashing is the situation that occurs when a system gets diverted from concentrating on achieving useful work to concentrating on dealing with requests for work. The system goes into an overload mode.	Psychologists have measured human productivity as a function of the rate of requests coming to a person. As the request rate rises, productivity increases towards a saturation limit. If the request rate rises beyond a critical threshold, productivity will drop suddenly and become much less than a person's	Storage systems are subject to thrashing. Operating systems use multiprogramming to load multiple processes into main (top level) memory so that when one stopped for an UP operation, the processor	Storage systems are subject to thrashing. Operating systems use multiprogramming to load multiple processes into main (top level) memory so that when one stopped for an UP operation, the processor

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	saturation limit. The critical threshold is not predictable. The explanation is that the brain gets working so hard to process incoming requests that it can no longer attend to the work of answering the requests. This is called "information overload" and that a person in this state is "thrashing". Storage systems are subject to thrashing. Cheese counter analogy - one server, issuing tickets to customers at the same time as trying to serve cheese. Denial of Service (DoS) attack. Distributed Denial of Service (DDoS) attack. Because Web sites are built to handle a lot of traffic, it can take millions of simultaneous communications requests to have enough affect on the performance of the server for an attack. In a DDoS attack, tens of thousands or even millions of computers are used to send traffic to the target site all at the same time and repeatedly. As Sophos' Graham Cluley wrote on his blog: "It's a bit like 15 fat men trying to get through a revolving door at the same time--nothing can move." Botnet - hijacked PCs are used in a	could switch to another. A share of memory is assigned to each user. When the share is smaller than the user's program, the program will be forced to stop from time to time to request a block transfer from the disk system. Each incoming block displaces an already-loaded block, which must be recalled in the future. As the number of logged-in users increases, the individual shares of memory become smaller and the recall traffic becomes more intense. Eventually almost all the user processes are waiting in queue for secondary memory system to respond to their call requests. from the user standpoint, the system appears to stop processing. Web server - Get / requests cause a child process to be spawned or a new thread to be created. Denial of Service attacks and Distributed Denial of Service attacks. Cyber terrorism.	could switch to another. A share of memory is assigned to each user. When the share is smaller than the user's program, the program will be forced to stop from time to time to request a block transfer from the disk system. Each incoming block displaces an already-loaded block, which must be recalled in the future. As the number of logged-in users increases, the individual shares of memory become smaller and the recall traffic becomes more intense. Eventually almost all the user processes are waiting in queue for secondary memory system to respond to their call requests. from the user standpoint, the system appears to stop processing. Solution - measure working sets, limit multiprogramming to processes whose working sets can be fully loaded. Multithreading, dlls help reduce overheads in multiprogramming (multiuser, multitasking environments). Web server - Get / requests cause a child process to be spawned or a new thread to be created. Denial of Service attacks and

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	DDoS attack. The individual computers are called "bots," "zombies" or "slaves" and are controlled remotely by the "master" attacker. The attacker relays instructions to the bots via a command-and-control server, typically using IRC (Internet Relay Chat). Botnets are also used to distribute spam.		Distributed Denial of Service attacks. Cyber terrorism.
3.10.2. Thrashing has been observed in many other contexts where multiple processes contend for a shared resource and the protocol that they use to determine who goes next has overhead that increases with the number of contenders. Eventually the overhead of contention resolution reduces the contender's capacity to do work.	 Distributed Denial of Service.	Web servers have also suffered from this problem - overloaded with requests	Web servers have also suffered from this problem - overloaded with requests causing response times to collapse. Solution is to distribute the user load across mirror servers in a way that did not force any of them into thrashing (mirror servers operate below thrashing threshold). ALOHA packet-radio system that extended the ARPANET packet switching idea to radio communications suffered from thrashing. Potential in local area networks and mobile phone networks for thrashing when several network stations attempt to send at the same time or several mobile phones in the same cell attempt to send on the same frequencies at the same time. Solution - CSMA/CD protocol for Ethernets and mobile phone networks. Ethernet switches reduce size of

Principle	Key Stage 3	Key Stage 4	Key Stage 5
		causing response times to collapse. Solution is to distribute the user load across mirror servers in a way that does not force any of them into thrashing (mirror servers operate below thrashing threshold).	collision domains
3.11 ALL COMMUNICATIONS TAKE PLACE IN STORAGE SYSTEMS			
3.11.1. Storage is essential for computation: all representations (of data and instructions) and their states must be held in a medium as the computation proceeds.	Non-volatile memory - paper - magnetic media - optical media - solid state - electronic paper Secondary storage devices and capacities Magnetic disk - PATA and SATA Hard Disk Drives HDD Compact disk - CD-ROM - CD-R - CD-RW Digital Versatile/Video Disk	Non-volatile memory - paper - magnetic media - optical media - solid state - electronic paper Secondary storage devices, principles of operation and capacities Magnetic disk Compact disk - CD-ROM - CD-R - CD-RW Digital Versatile/Video Disk	Non-volatile memory - paper - magnetic media - optical media - solid state - electronic paper Secondary storage devices, principles of operation, capacities, access times, data transfer rates Magnetic disk Compact disk - CD-ROM - CD-R - CD-RW Digital Versatile/Video Disk

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	- DVD-ROM - DVD-R - DVD-RW High Definition/Density Digital Versatile/Video Disk - HD-DVD ROM - HD-DVD-R - HD-DVD-RW - Blu-ray ROM - Blu-ray R - Blu -ray RW	- DVD-ROM - DVD-R - DVD-RW High Definition/Density Digital Versatile/Video Disk - HD-DVD ROM - HD-DVD-R - HD-DVD-RW - Blu-ray ROM - Blu-ray R - Blu -ray RW	- DVD-ROM - DVD-R - DVD-RW High Definition/Density Digital Versatile/Video Disk - HD-DVD ROM - HD-DVD-R - HD-DVD-RW - Blu-ray ROM - Blu-ray R - Blu -ray RW
3.11.2. A storage system consists of various storage devices of different media storage capacities and access times, with methods for storing and retrieving data.	**Non-volatile memory continued** Solid-state memory - ROM - PROM - EEPROM - Flash □ NOR □ NAND ● Single Level Cell (one bit per cell), SLC ● Multiple Level Cell (more than one bit per cell) MLC	**Non-volatile memory continued** Solid-state memory - ROM - PROM - EEPROM - Flash □ NOR □ NAND ● Single Level Cell (one bit per cell), SLC ● Multiple Level Cell (more than one bit per cell) MLC	**Non-volatile memory continued** Magnetic tape recording medium - analogue recording - digital recording Magnetic tape types - open reel - cassette - cartridge □ DAT/DDS, DLT and LTO Solid-state memory - ROM

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	□ USB Memory sticks □ Flash memory cards for mobile phones, digital cameras, PDAs, GPS Sat-Nav systems □ Flash memory SATA solid state drives Comparative capacities of each of the storage system types Bit, Byte, KiloByte (KB) 1024, MegaByte (MB) 1024x1024, GigaByte (GB) 1024x1024x1024, TeraByte (TB) 1024x1024x1024x1024, PetaByte (PB) 1024x1024x1024x1024x1024 Endurance, data retention, robustness	□ USB Memory sticks □ Flash memory cards for mobile phones, digital cameras, PDAs, GPS Sat-Nav systems □ Flash memory SATA solid state drives Comparative capacities of each of the storage system types Comparative access times and data transfer rates Endurance, data retention, robustness	- PROM - EEPROM - Flash □ NOR □ NAND ● Single Level Cell (one bit per cell), SLC ● Multiple Level Cell (more than one bit per cell) MLC □ USB Memory sticks □ Flash memory cards for mobile phones, digital cameras, PDAs, GPS Sat-Nav systems □ Flash memory SATA solid state drives Comparative capacities of each of the storage system types Comparative access times and data transfer rates Endurance, data retention, robustness
3.11.3. A storage system consists of various storage devices of different media storage capacities and access times, with methods for storing and retrieving data.	**Structure of HDD** - Multiple platters - Surface - Sector - Track	**Structure of HDD** - Multiple platters - Surface - Sector - Track	**Structure of HDD** - Multiple platters - Surface - Sector - Track

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	- Read/Write head mounted on moveable arm - Platters rotate at 5000 - 7200 RPM when reading/writing - Platters still rotate but at slower rate when not in use - Logical Blocks **Structure of SSD** - Same form factor as HDD (2.5" and 1.8") - Interfaces exactly as would a HDD - Logical blocks - No moving parts SSD more robust than HDD and of lower power consumption	- Read/Write head mounted on moveable arm - Platters rotate at 5000 - 7200 RPM when reading/writing - Platters still rotate but at slower rate when not in use - Logical Blocks - Cylinder **Structure of SSD** - Same form factor as HDD (2.5" and 1.8") - SATA interface (externally modelled on HDD) - Logical blocks - No seek, latency low because simultaneous random access to multiple blocks - No moving parts SSD more robust than HDD and of lower power consumption	- Read/Write head mounted on moveable arm - Platters rotate at 5000 - 7200 RPM when reading/writing - Platters still rotate but at slower rate when not in use - Logical Blocks - Cylinder **Structure of SSD** - Same form factor as HDD (2.5" and 1.8") - SATA interface (externally modelled on HDD) - Logical blocks - No seek, latency low because simultaneous random access to multiple blocks - No moving parts SSD more robust than HDD and of lower power consumption Seek time, Latency for HDD, SSD, Optical media
3.11.4. A storage system consists of various storage devices of different media storage capacities and access	**Structure of optical media devices** - One continuous track that spirals outwards from centre of disk	**Structure of optical media devices** - One continuous track that spirals outwards from centre of disk	**Structure of optical media devices** - One continuous track that spirals outwards from centre of disk

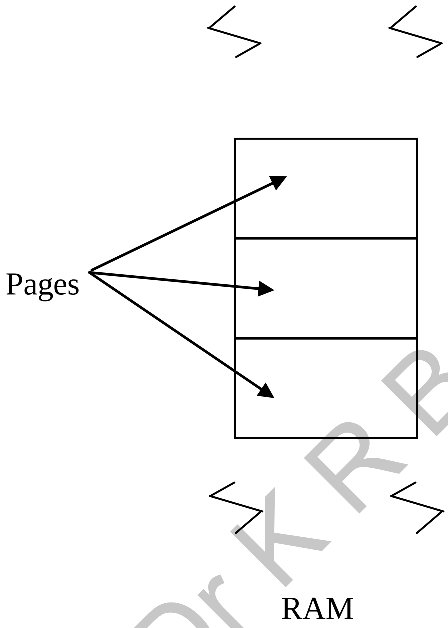
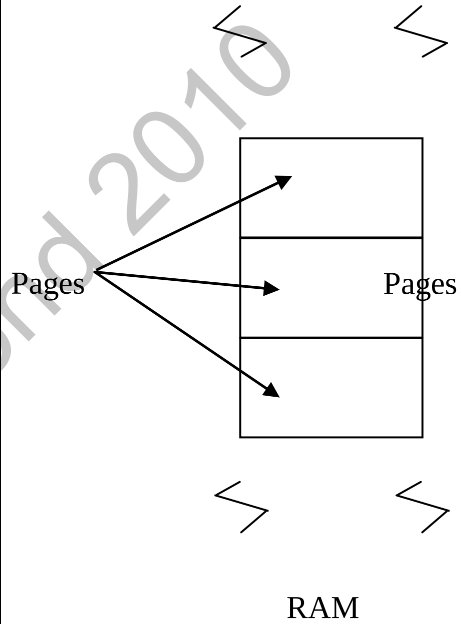
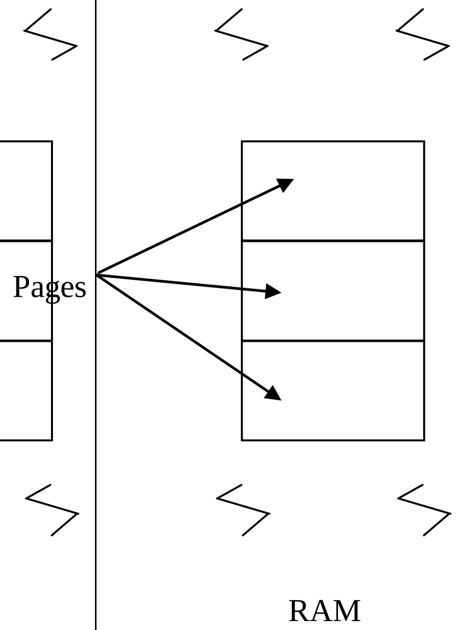
Principle	Key Stage 3	Key Stage 4	Key Stage 5
times, with methods for storing and retrieving data.	- laser used to read and write - Amount of light reflected determines whether 0 or 1 read - Read/Write surface coated with a protective layer	- laser used to read and write - Amount of light reflected determines whether 0 or 1 read - Writing a bit means affecting the way that the medium reflects light - Read/Write surface coated with a protective layer - Can use multiple layers within medium for writing to/reading from to increase capacity - packing more spirals across disk increases capacity - packing more bits per cm along track increases capacity	- laser used to read and write - Amount of light reflected determines whether 0 or 1 read - Writing a bit means affecting the way that the medium reflects light - Read/Write surface coated with a protective layer - Can use multiple layers within medium for writing to/reading from to increase capacity - packing more spirals across disk increases capacity - packing more bits per cm along track increases capacity - use of lasers of different wavelength and power - the different technologies used to encode a bit
3.11.5. The arrangement of data in storage significantly affects the resolution time of an algorithm.	For example, if the data are N items arranged in a random order, the time to find a particular item will be proportional to N (just count number of steps). But if the N items are arranged in increasing order, the search time drops to order log N (just count average number of steps). Most of complexity theory assumes that the data are laid arranged in a “flat” medium that stores and retrieves any	For example, if the data are N items arranged in a random order, the time to find a particular item will be proportional to N. But if the N items are arranged in increasing order, the search time drops to order log N. Most of complexity theory assumes that the data are laid arranged in a “flat” medium that stores and retrieves any item in the same	For example, if the data are N items arranged in a random order, the time to find a particular item will be proportional to N. But if the N items are arranged in increasing order, the search time drops to order log N. Most of complexity theory assumes that the data are laid arranged in a “flat” medium that stores and

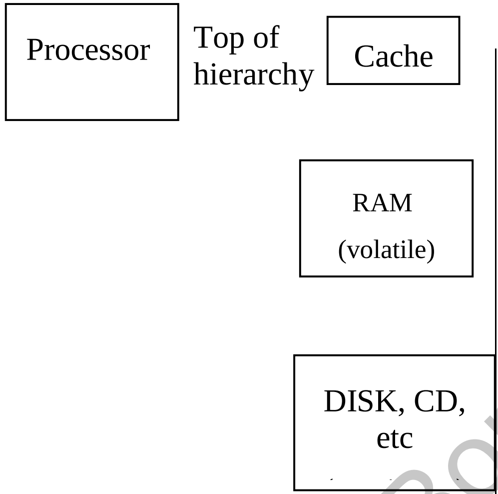
Principle	Key Stage 3	Key Stage 4	Key Stage 5
	item in the same amount of time. For flat storage, counting an algorithm's steps is a reasonable way to assess its running time.	amount of time. For flat storage, counting an algorithm's steps is a reasonable way to assess its running time.	retrieves any item in the same amount of time. For flat storage, counting an algorithm's steps is a reasonable way to assess its running time.
3.11.6. A flat storage system is a model in which every item of storage has the same access time. In such a medium, it is reasonable to associate resolution time of an algorithm with the number of steps it takes to complete a computation.	RAM is flat - Random Access Memory is named because order in which memory cells accessed should not affect total access time	RAM is flat - Random Access Memory is named because order in which memory cells accessed should not affect total access time	RAM is flat - Random Access Memory is named because order in which memory cells accessed should not affect total access time
3.11.7. Storage systems are not flat. **These storage systems have hierarchical structures. Hierarchical structure** **significantly affects performance.**	Every storage system is arranged as a hierarchy with fast-access at the top and slow-access at the bottom. Amazon.co.uk, the online bookstore, keeps its popular books in warehouse ready to ship. Less popular books are stored by their printers who fulfil orders forwarded from Amazon.co.uk. A storage system consists of a collection of devices of different media, each of its own price, access time and transfer rate. The top of the hierarchy is called the "Main Store" or "Main Memory" or "computational store". It holds the data directly accessible to the processor. This is RAM local to the processor. Many storage systems (such as the	Every storage system is arranged as a hierarchy with fast-access at the top and slow-access at the bottom. Amazon.co.uk, the online bookstore, keeps its popular books in warehouse ready to ship. Less popular books are stored by their printers who fulfil orders forwarded from Amazon.co.uk. A storage system consists of a collection of devices of different media, each of its own price, access time and transfer rate. The top of the hierarchy is called the "Main Store" or "Main Memory" or "computational store". It holds the data directly accessible to the processor. This is	Every storage system is arranged as a hierarchy with fast-access at the top and slow-access at the bottom. Amazon.co.uk, the online bookstore, keeps its popular books in warehouse ready to ship. Less popular books are stored by their printers who fulfil orders forwarded from Amazon.co.uk. A storage system consists of a collection of devices of different media, each of its own price, access time and transfer rate. The top of the hierarchy is called the "Main Store" or "Main Memory" or "computational store". It holds the data directly accessible to the

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	Internet) are shared among many users. each pulls data from the storage system into its local RAM. Thus each user sees a unique hierarchy, with his/her own RAM at the top. The rest of a storage hierarchy is called "secondary store" or "long-term store" or "backing store". They are persistent media, retaining indefinitely until they are explicitly erased, e.g. magnetic disks.	RAM local to the processor. Many storage systems (such as the Internet) are shared among many users. each pulls data from the storage system into its local RAM. Thus each user sees a unique hierarchy, with his/her own RAM at the top. The rest of a storage hierarchy is called "secondary store" or "long-term store" or "backing store". They are persistent media, retaining indefinitely until they are explicitly erased, e.g. magnetic disks.	processor. This is RAM local to the processor. Many storage systems (such as the Internet) are shared among many users. each pulls data from the storage system into its local RAM. Thus each user sees a unique hierarchy, with his/her own RAM at the top. The rest of a storage hierarchy is called "secondary store" or "long-term store" or "backing store". They are persistent media, retaining indefinitely until they are explicitly erased, e.g. magnetic disks.
3.11.8. The arrangement of data across devices of a non-flat storage system significantly affects the resolution time of an algorithm. For example, if the flat part (RAM) can hold only a limited amount of data for a computation, then if additional data needs to be fetched from non-flat storage (disk) before the computation can be completed then memory load operations must be performed. These are typically much slower than processor instructions.	Notice how long it takes to move to the end of a very long word processed document when available memory is low and the tail of the document has to be fetched from disk Access time for RAM ~ 10^{-9} seconds Access time for hard disk ~ 10^{-2} seconds.	Notice how long it takes to add 1000000 1s fetched from a sequential file compared with when stored in an array.	Consider the situation when a straightforward arithmetic problem has to use non-flat memory. For example, multiplying two N x N matrices in a flat memory takes time proportional to N^3. The algorithm needs $3N^2$ memory cells to hold the two input and one output matrices. Suppose we have matrices so large that the flat part of memory can only hold a single row or column from each input matrix. In this system, each output element will require a load operation for a row or column of an input matrix, for a total of N^2 loads. Because the load operations can take much longer than individual instructions, the total

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			loading cost can easily dwarf the computational cost. A hypothetical weather forecasting problem using a grid of 4K ($N=2^{12}$) points on a side. The computation for a matrix multiply would then be about 2^{36} (N^3)operations. A gigaops ($10^9 = 2^{30}$ operations/sec) desktop computer would take about $2^6 = 64$ seconds for the computation. However, in the constrained memory, it would need about $2^{24} = 10^{11}$ load operations. If the loads came from a disk that takes 10 milliseconds per row or column, the total load time would be about 10^9 seconds, or about 30 years! The cost of loading far exceeds the cost of computing.

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.12 STORAGE SYSTEMS COMPRISE HIERARCHIES WITH VOLATILE (FAST) STORAGE AT THE TOP AND PERSISTENT (SLOWER) STORAGE AT THE BOTTOM			
3.12.1. Storage devices are volatile or persistent (non-volatile).	Volatile means that the stored data can be maintained only if energy is constantly supplied to the medium (e.g. RAM). Persistent means that the stored data are inscribed in the medium and will remain there without additional energy until erased (e.g. hard disk). Volatile media are fast; persistent media are much slower but also cheaper. Most computing systems use volatile storage (e.g. RAM) for holding data accessed directly by the processor (or any processing elements) and persistent storage (e.g. hard disk) for holding data over indefinite periods independent of any processor.	Volatile means that the stored data can be maintained only if energy is constantly supplied to the medium (e.g. RAM). Persistent means that the stored data are inscribed in the medium and will remain there without additional energy until erased (e.g. hard disk). Volatile media are fast; persistent media are much slower but also cheaper. Most computing systems use volatile storage (e.g. RAM) for holding data accessed directly by the processor (or processing elements) and persistent storage (e.g. hard disk) for holding data over indefinite periods independent of any processor.	Volatile means that the stored data can be maintained only if energy is constantly supplied to the medium (e.g. RAM). Persistent means that the stored data are inscribed in the medium and will remain there without additional energy until erased (e.g. hard disk). Volatile media are fast; persistent media are much slower but also cheaper. Most computing systems use volatile storage (e.g. RAM) for holding data accessed directly by the processor (or any processing elements) and persistent storage (e.g. hard disk) for holding data over indefinite periods independent of any processor.

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.12.2. Data are organised into blocks, which are units of storage and transfer. Blocks all of the same size are called "pages". Blocks of different sizes are called "segments". **The generic term "object" will be used for a block of storage.**	RAM 	RAM 	RAM 
3.12.3. A storage hierarchy has the fastest (and most expensive) devices at the top and progressively slower (and less expensive) devices farther down. A typical hierarchy has levels for CACHE, RAM and DISK. A master copy of data resides in the persistent levels.		The cost per bit of main store is much higher than secondary store. RAM is approximately 100 times more expensive per bit than disk so a system tends to have much more disk storage space than RAM. The disk has access time around 10^{-3} second compared to the RAM's 10^{-9} second, a gap of 10^{6} RAM cycles for	The cost per bit of main store is much higher than secondary store. RAM is approximately 100 times more expensive per bit than disk so a system tends to have much more disk storage space than RAM. The disk has access time around 10^{-3} second compared to the RAM's 10^{-9} second, a gap of 10^{6} RAM

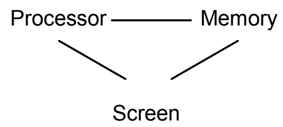
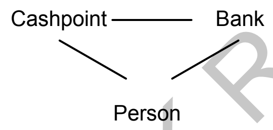
Principle	Key Stage 3	Key Stage 4	Key Stage 5
An UP operation pulls a copy of an object to the top of the hierarchy. A DOWN operation pushes a copy of a modified object down the hierarchy, updating the master when it reaches that level.	 Many local RAMs are augmented with a cache, a small amount of ultra-fast expensive RAM. The cache can run at the speed of the processor but the RAM cannot. The cache therefore allows the processor to run faster.	one disk access. A RAM cycle is one RAM access. If 99.99% of processor references land on objects in RAM, 0.01% land on objects in secondary store. The effective memory access time = $(1)x(0.9999) + (10^6)x(0.0001) = 100$ RAM cycles. Therefore, the processor runs at an average rate of 1/100 of the actual RAM speed even though most of the time it is accessing RAM. UP and DOWN operations can be automated. Typically UP operations are issued on demand - when processor attempts to access object that is not in top-level memory. When confidence prediction is high, UP operations can be issued well before the object is needed. A replacement policy determines which object in CACHE or RAM must be moved down to make way for an up-coming object.	cycles for one disk access. A RAM cycle is one RAM access. If 99.99% of processor references land on objects in RAM, 0.01% land on objects in secondary store. The effective memory access time = $(1)x(0.9999) + (10^6)x(0.0001) = 100$ RAM cycles. Therefore, the processor runs at an average rate of 1/100 of the actual RAM speed even though most of the time it is accessing RAM. Virtual memory

Principle	Key Stage 3	Key Stage 4	Key Stage 5

Learning outcomes for Key Stages 3 to 5 Computing. Strand: SYSTEMS

Learners should know and understand……

4. SYSTEMS PRINCIPLES			
4.1 **Systems thinking** is the process of understanding how things influence one another within a whole			
4.1.1. A system is a dynamic and complex whole, interacting as a structured functional unit	Emergent behaviour Simulation – SimCity In modern computing we build and analyse huge systems, equal in complexity to many systems found in nature . e.g. an ecology. So in computing, as in natural science, there must be many levels of description. Computer science has its organisms, its molecules and its elementary particles . its biology, chemistry and physics: **LEVELS OF DESCRIPTION** **NATURAL SCIENCE** — **COMPUTER SCIENCE** Biology ORGANISMS — Databases, Networks, . .. Chemistry MOLECULES — Metaphors of programming Physics PARTICLES (ELEMENTS) — Primitives of programming	Emergent behaviour GAIA hypothesis	Emergent behaviour Simulation

Principle	Key Stage 3	Key Stage 4	Key Stage 5
4.1.2 Modelling systems by interfaces	Open systems are defined as systems that can be modelled by interaction machines, while closed systems are modelled by algorithms. INSIDE  OUTSIDE 	Open systems are defined as systems that can be modelled by interaction machines, while closed systems are modelled by algorithms.	System specification by parts and its forms of behaviour, e.g. Requirements of an airline reservation system may be specified by the set of all interfaces (modes of use) it should support. Airline reservation system consists of multiple instances of the following kind of interfaces: Travel agents: making reservations on behalf of clients Passengers: making direct reservations Airline desk employees: making enquiries on behalf of clients Flight attendants: aiding passengers during the flight itself Accountants: aiding and checking financial transactions Systems builders: developing and modifying the system Harnesses - Open - Closed Component Object Model People as collections of interfaces “Thinking” better described by interaction machines with multiple interfaces rather than by Turing machines

Principle	Key Stage 3	Key Stage 4	Key Stage 5

Learning outcomes for Key Stages 3 to 5 Computing. Strand: DESIGN

Learners should know and understand……

5. DESIGN PRINCIPLES			
5.1 ABSTRACTION, INFORMATION HIDING AND DECOMPOSITION ARE COMPLEMENTARY ASPECTS OF MODULARITY			
5.1.1. Modularity is a process of dividing a large system into a hierarchy of aggregates (modules) that interact across precisely defined interfaces.		Hierarchy chart of procedures	Hierarchy chart of procedures Structure chart Separately compilable units
5.1.2. Abstraction means to define a simplified version of something and to state the operations (functions) that apply to it. By bringing out the essence and suppressing detail, an abstraction offers a simple set of operations that apply to all the cases. In a hierarchy, an abstraction corresponds to an aggregate; forming a hierarchy is a process of abstraction.	Kinds of abstraction used in computation - Generalisation or classification □ Classification: Rising up the hierarchy means going from specific examples to categories that examples belong to □ Generalisation: Can be applied to problem solving by identifying a principle shared among solutions to different	Abstractions in classical science are mostly explanatory -- they define fundamental laws and describe how things work. Computer science abstractions do more: they define computational objects, and they perform actions. For example, the bit (0 or 1) is an abstraction of all sorts of media that rely on two states to store or transmit information -- pits and	Abstractions in classical science are mostly explanatory -- they define fundamental laws and describe how things work. Computer science abstractions do more: they define computational objects, and they perform actions. For example, the bit (0 or 1) is an abstraction of all sorts of media that rely on two states to store or transmit information -- pits and

Principle	Key Stage 3	Key Stage 4	Key Stage 5
Abstraction is one of the most fundamental powers of the human brain.	problems, e.g. Pigeon Hole principle - Representation abstraction or problem abstraction or reduction □ Details are removed until it becomes possible to represent the problem in a way that is possible to solve	peaks on a CD, magnetized patches on a disk, voltages on a wire. A file (a named sequence of bits) is a common abstraction representing text documents, graphs, spreadsheets, images, movies, sounds, directories, and more. A file system provides create, delete, open, close, read, and write operations that work on any file. Any program's output (already represented as bits) can be stored in a file. The file system does not have to understand the differences between file formats assigned by applications -- it just stores and retrieves the bits.	peaks on a CD, magnetized patches on a disk, voltages on a wire. Kinds of abstraction used in computation - Generalisation or classification □ Classification: Rising up the hierarchy means going from specific examples to categories that examples belong to □ Generalisation: Can be applied to problem solving by identifying a principle shared among solutions to different problems, e.g. Pigeon Hole principle - Representation abstraction or problem abstraction or reduction □ Details are removed until it becomes possible to represent the problem in a way that is possible to solve e.g. map colouring, mobile phone frequency allocation, fish allocation to fish tanks all reduce to a graph representation that can be solved by the same algorithm
5.1.3. Information hiding means to hide the details of an implementation so that users do not see them. It protects	Object-oriented programming • Greenfoot • Java-based • GameMaker	Object-oriented programming • Greenfoot • Blackfoot	Information hiding prevents users of a file system from seeing disks, disk drivers, records, index tables, disk addresses, buffers, caches, open file control blocks, and RAM copies

Principle	Key Stage 3	Key Stage 4	Key Stage 5
against errors caused by changes in the details that do not concern users. It is a policy that supports abstraction by preventing users of the abstraction from gaining access to the suppressed details behind the abstraction. In a hierarchy, it is a decision to hide the component structure of an aggregate, allowing that structure to be rearranged without changing the behaviour of the aggregate. A software module implements a software function by hiding internal details behind a simple interface.	• Blackfoot • Object Pascal-based		of files. This benefits the user in two ways: (1) the user never makes assumptions about the details, simplifying the user's task, and (2) the maintainer of the abstraction can change or improve any of the details without forcing any user to change anything. Object-oriented programming
5.1.4. Decomposition means to subdivide a large problem into components that can be designed separately and then assembled into the full system. In a hierarchy, identifying the components of an aggregate is an act of decomposition. A module is an abstraction of the components that compose it.		Structured programming - stepwise refinement - Top down design and development - Procedure/Function interfaces - Stubs and testing with stubs	Structured programming - stepwise refinement - Top down design and development - Procedure/Function interfaces – Stubs and testing with stubs

Principle	Key Stage 3	Key Stage 4	Key Stage 5
5.2 The four base principles of software design are hierarchical aggregation, levels, virtual machines, and objects.			
5.2.1. Hierarchical aggregation means that objects (components) consist of interconnected groups of smaller objects and are themselves components of larger objects. The principles of abstraction, information hiding, and decomposition all flow from this one.	Interact with an object as a unit without being concerned about the individual parts constituting it. When you look inside you focus on the interactions among components without concern about what is going on in the external environment. Hierarchical aggregation is common in nature: - quarks, electrons and protons, atoms, molecules, materials, planets, solar systems, galaxies - DNA, genes, cells, organs, nervous systems, plants, animals and social systems. Principle of locality is a consequence of hierarchical aggregation	Interact with an object as a unity without being concerned about the individual parts constituting it. When you look inside you focus on the interactions among components without concern about what is going on in the external environment. Hierarchical aggregation is common in nature: - quarks, electrons and protons, atoms, molecules, materials, planets, solar systems, galaxies - DNA, genes, cells, organs, nervous systems, plants, animals and social systems. Principle of locality is a consequence of hierarchical aggregation	Interact with an object as a unity without being concerned about the individual parts constituting it. When you look inside you focus on the interactions among components without concern about what is going on in the external environment. Hierarchical aggregation is common in nature: - quarks, electrons and protons, atoms, molecules, materials, planets, solar systems, galaxies - DNA, genes, cells, organs, nervous systems, plants, animals and social systems. Principle of locality is a consequence of hierarchical aggregation
5.2.2. Levels means to organize the functions of a system into a hierarchy with the constraint that the higher-up functions can only call lower-down functions. Virtual machines means to	Levels is a form of aggregation that stratifies the levels abstraction. All objects in the same level are treated as peers with respect to how they contribute to the higher levels. For example, physical materials can be	Levels is a form of aggregation that stratifies the levels abstraction. All objects in the same level are treated as peers with respect to how they contribute to the higher levels. The levels principle is derived from	Levels is a form of aggregation that stratifies the levels abstraction. All objects in the same level are treated as peers with respect to how they contribute to the higher levels. File transfer protocol, FTP, for

Principle	Key Stage 3	Key Stage 4	Key Stage 5
create simulations of logical computing machines and use them to solve the problem. Virtual machines can be hierarchically nested.	seen as assemblies of molecules, molecules as assemblies of atoms, atoms as assemblies of electrons and protons and neutrons and so on.	traditional science to explain complex systems by dividing into levels that can be understood in terms of their own abstractions. For example, physical materials can be seen as assemblies of molecules, molecules as assemblies of atoms, atoms as assemblies of electrons and protons and neutrons and so on. In computing, levels means to organise the functions of a large system into a hierarchy. Functions in a given level are components (decompositions) of a function at a higher level. Functions in a given level may invoke functions in a lower level but never a function in a higher level. The levels structure also facilitates testing. The lowest level is tested first and so on.	example is built on several lower layers including IP protocol, routing protocol, data link protocol and physical signal protocol. The levels principle applies nature's learning to complex networked software systems. The levels principle is derived from traditional science to explain complex systems by dividing into levels that can be understood in terms of their own abstractions. For example, physical materials can be seen as assemblies of molecules, molecules as assemblies of atoms, atoms as assemblies of electrons and protons and neutrons and so on. In computing, levels means to organise the functions of a large system into a hierarchy. Functions in a given level are components (decompositions) of a function at a higher level. Functions in a given level may invoke functions in a lower level but never a function in a higher level. This structure simplifies the proof of correctness which can be considered one level at a time. The lowest level, which does not call lower-level functions,

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			is proved first and so on. The levels structure also facilitates testing. The lowest level is tested first and so on. Levels of an operating system - Kernel - Device drivers - Processor management - Memory management - File management - I/O management - User interface(command line and GUI) and application programming interface
5.2.3. Virtual machines means to create simulations of logical computing machines and use them to solve the problem. Virtual machines can be hierarchically nested.		A virtual machine is a simulation of one computer by another. The idea traces back to Alan Turing's Universal Machine. The simulation principle behind it says that a more complex system can be simulated from less complex components. Java Virtual Machine	A virtual machine is a simulation of one computer by another. The idea traces back to Alan Turing's Universal Machine. The simulation principle behind it says that a more complex system can be simulated from less complex components. Java Virtual Machine Windows Vista Windows hosted on Apple

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			machines Virtualisation Process - program in execution on a virtual machine
5.2.4. Objects means to encapsulate a data structure and its applicable functions in a single package.		Software programs can transform a computer into any other virtual machine. For this reason, it has become common to say that software creates abstractions that execute and perform actions. The designers job is to find a decomposition of the original problem into a hierarchy of abstractions that solve the problem. These are called objects, e.g. a file	Software programs can transform a computer into any other virtual machine. For this reason, it has become common to say that software creates abstractions that execute and perform actions. The designers job is to find a decomposition of the original problem into a hierarchy of abstractions that solve the problem. These are called objects, e.g. a file
5.3 Design principles are conventions for planning and building correct, fast, fault tolerant, and fit software systems			
5.3.1. Design means two things: architecture and process. Architecture is a division of a system into components, their interactions and their layout. Process is the steps producing an architecture.	Hardware architecture Software architecture	Hardware architecture Software architecture	Hardware architecture Software architecture
5.3.2. Design process is adopted from engineering **1. Requirements**		Waterfall model Spiral model	Waterfall model Spiral model

Principle	Key Stage 3	Key Stage 4	Key Stage 5
2. Specifications **3. Prototype** **4. Testing**		Requirements analysis Requirements specification Design specifications Prototyping Testing strategies: - Bottom-up testing - Top-down testing - Dry running - White box testing - Black box testing Test plan Test cases Choice of test data - Normal data - Boundary values - Erroneous values - Extreme values - Exceptional data	Agile development Requirements analysis Requirements specification Design specifications Prototyping Testing strategies: - Bottom-up testing - Top-down testing - Mixed-level testing - Dry running - White box testing - Black box testing - Unit testing - Integration testing - Regression testing - Model-based testing - System testing - Alpha testing - Beta testing - Acceptance testing Test plan Test cases

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			Choice of test data - Normal data - Boundary values - Erroneous values - Extreme values - Exceptional data
5.3.3. Four main criteria for good design: **1. Correctness** **2. Speed** **3. Tolerance** **4. Fit**			Babbage machine - to overcome errors in hand-calculated numerical tables of logarithms, etc used to compute the orbits of planets and positions of stars for navigators.
5.3.4. Correctness means that the software system provably meets precise specifications. Correctness is challenging because of the difficulty of getting precise specifications for complex systems and the computational intractability of formal proofs for large systems	Mistakes in programs: Lexical errors Syntax errors Logical errors - calculation of incorrect values for some inputs - entering infinite loops for some inputs Decomposing complex systems into small components using subroutines Instructions for creating a kite	Mistakes in programs: Lexical errors Syntax errors Logical errors - calculation of incorrect values for some inputs - entering infinite loops for some inputs Decomposing complex systems into small components using subroutines	Babbage machine - to overcome errors in hand-calculated numerical tables of logarithms, etc used to compute the orbits of planets and positions of stars for navigators. Use of induction to prove correctness of a simple algorithm, e.g. SumNaturalNumbers (n) Invariants Decomposing complex programs into small components enabling specifications to be simpler and easier to check:

Principle	**Key Stage 3**	**Key Stage 4**	**Key Stage 5**
			- Call and return subroutines to allow programmers to encapsulate a complex routine behind a simple interface
5.3.5. Speed means that the system completes tasks within acceptable time limits.			Computational complexity dominates performance of algorithms on ideal computers - flat memories and fast processors. Algorithms are classified by order of growth of their computation with size of input. However, knowledge that an algorithm is in one of these categories only allows ranking of the algorithm relative to others of the same or different orders. In reality software is never a single algorithm run alone on a single computer. Software is always a mixture of algorithms of different running times executed on machines shared with other users. User processes (also called jobs) compete for limited processor, memory disk and networking resources. Contention for these resources adds delay above and beyond what the software would require if it were able to operate alone on a machine. Performance , therefore, is almost always formulated in a systems context, where the analyst must provide answers such as:

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			1. What is the throughput (jobs per second)? 2. What is the response time to a user request? 3. What are the bottlenecks? 4. How much capacity is needed to meet throughput and response time objectives?
5.3.6. Fault tolerance means that the software and host systems can continue to function despite small errors and will refuse to function in the case of large errors.		Software creates virtual worlds. These are highly sensitive to errors. A single bit changed in a program can drastically change the algorithm represented by the program. If this controls some real world process, e.g. rocket flight, a disaster can result. In the real world, physical systems obey continuum laws that guarantee that a small change in one variable produces a small corresponding change in other (dependent) variables. Via natural feedback systems, the error can be easily corrected. Thus the system can naturally tolerate a small error. Many biological systems, including human and animal immune systems, contain self-repair mechanisms that respond to errors through feedback and correct them.	Error confinement: - Limit the number of objects that an error can influence before it is detected Error recovery - Remove the error and restore an error-free condition Error recovery is easier when errors are confined Static checks in the software: - Type checks by compiler Dynamic checks in the software - Array bounds checking - Exception trapping and handling Dynamic checks in the host environment in which software runs: - Checks on data tags: a handle

Principle	Key Stage 3	Key Stage 4	Key Stage 5
		Static checks in the software: - Type checks by compiler Dynamic checks in the software - Array bounds checking - Exception trapping and handling Fault-tolerant computer systems	passed to any file system procedure must be tagged "file" - Checks on memory references Fault-tolerant computer systems - RAID - Systems are replicated
5.3.7. Fitness means that the dynamic behaviour of the system aligns with the environment of its use.	The principles of fitness are: 1. Every technology enables a practice; the software engineer is designing a new practice or aligning with an existing practice. 2. Fit is most easily achieved when the practice aligns with pre-existing practice or is intuitive and obvious Examples of a successful fits: ATM system - automated the teller without changing any banking practice. It was easy for customers to use. They simply did what they always did and it worked. The ATM is an excellent fit between a machine and the standard practices of its user community. Google Amazon.co.uk Apple iphone	The principles of fitness are: 1. Every technology enables a practice; the software engineer is designing a new practice or aligning with an existing practice. 2. Fit is most easily achieved when the practice aligns with pre-existing practice or is intuitive and obvious Examples of a successful fits: ATM system - automated the teller without changing any banking practice. It was easy for customers to use. They simply did what they always did and it worked. The ATM is an excellent fit between a machine and the standard practices of its user community. Google Amazon.co.uk Apple iphone	The principles of fitness are: 1. Every technology enables a practice; the software engineer is designing a new practice or aligning with an existing practice. 2. Fit is most easily achieved when the practice aligns with pre-existing practice or is intuitive and obvious Examples of a successful fits: ATM system - automated the teller without changing any banking practice. It was easy for customers to use. They simply did what they always did and it worked. The ATM is an excellent fit between a machine and the standard practices of its user community. Google Amazon.co.uk Apple iphone

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	Examples of an unsuccessful fits: VCR on-line help systems	Examples of an unsuccessful fits: VCR on-line help systems	Examples of an unsuccessful fits: VCR on-line help systems
5.4 Objects organize software into networks of shared entities that activate operations in each other by exchanging signals			
5.4.1. A software object is an abstraction of a data entity packaged together with the set of functions that operate on it. The user of an object signals the desired function; the object performs that function, updates its internal state, and returns an answer to the user.	Use of objects in a gaming environment e.g. GameMaker, Greenfoot, Blackfoot	Object-oriented programming Objects Object methods Object properties Invoking a method Instantiating an object	Object-oriented programming Objects Object methods Object properties Invoking a method Instantiating an object
5.4.2. Objects are defined as members of a class of similar objects, usually called a type. All objects of the same type have the same operations. **Types are defined in terms of higher-level types (hierarchies) so that an object can inherit properties from its immediately defining class and from all higher classes.**	Class of objects Simple gaming environment e.g. GameMaker, Greenfoot, Blackfoot	Classes Inheritance	Classes Inheritance Class diagram - Inheritance diagram Object diagram Aggregation/composition - fixed - variable

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			Aggregation diagram Polymorphism Association - Uses relationship
5.4.3. Objects have locks so that parallel processes can access them only one at a time (mutual exclusion).			To provide deterministic behaviour in multithreaded applications, locks are used to synchronise the concurrent activity of threads. Only one thread at a time is allowed to execute a region of code affecting the state of an object. There is a lock associated with each object. This can be demonstrated in a version of the Scratch programming language known as BYOB.
5.4.4. Objects have global handles so that they can be shared among many users.			COM and DCOM models Globally Unique Identifiers (GUIDs)
5.5 In a distributed system, it is more efficient to implement a function in the communicating applications than in the network itself (end-to-end principle)			
5.5.1. This claim, known as the end-to-end principle, is simply that the network itself does not know all the requirements of the applications that connect to		Consider any network in which data are moved from one machine to another. If we check the integrity of the bits at any point on the path, but not the absolute end, there is a possibility that an error can occur in	Consider any network in which data are moved from one machine to another. If we check the integrity of the bits at any point on the path, but not the absolute end, there is a possibility that an error can occur in

Principle	Key Stage 3	Key Stage 4	Key Stage 5
it; therefore any attempt to implement in the network a function used solely by the applications will be less efficient than implementing the function in the applications but not the network.		the final segment. Therefore, we must check that the bits at the end are the same as those at the beginning. This is called end-to-end error checking. The TCP (transport control protocol) of the Internet uses end-to-end checking to assure that files are transferred with 100% reliability. The source file is broken into packets, which are sequence-numbered and sent to the receiver. The receiver checks parity bits to detect corrupted packets, and it notes gaps in its list of received packets to detect missing packets. It sends acknowledgements to the sender that confirm the number of packets successfully received. Both sender and receiver have time-out alarms after which they resend packets for which there has been no acknowledgement. There is no checking in the Internet itself to be sure that packets have not been corrupted or lost. The Internet is simply a best-effort medium that tries to get packets delivered but may not succeed, and it does not retry. The end-to-end checking of the TCP makes file transfer reliable even though packet transfer is unreliable.	the final segment. Therefore, we must check that the bits at the end are the same as those at the beginning. This is called end-to-end error checking. The TCP (transport control protocol) of the Internet uses end-to-end checking to assure that files are transferred with 100% reliability. The source file is broken into packets, which are sequence-numbered and sent to the receiver. The receiver checks parity bits to detect corrupted packets, and it notes gaps in its list of received packets to detect missing packets. It sends acknowledgements to the sender that confirm the number of packets successfully received. Both sender and receiver have time-out alarms after which they resend packets for which there has been no acknowledgement. There is no checking in the Internet itself to be sure that packets have not been corrupted or lost. The Internet is simply a best-effort medium that tries to get packets delivered but may not succeed, and it does not retry. The end-to-end checking of the TCP makes file transfer reliable even though packet transfer is

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			unreliable.
Learning outcomes for Key Stages 3 to 5 Computing. Strand: COORDINATION ***Learners should know and understand……***			
6. COORDINATION PRINCIPLES			
6.1 A COORDINATION SYSTEM IS A SET OF AGENTS INTERACTING WITHIN A FINITE OR AN INFINITE GAME TOWARD A COMMON OBJECTIVE			
6.1.1. Agents can be humans or computational processes	Coordination is as fundamental in computation as it is in other parts of nature Coordination in nature - flocking behaviour • Amongst birds, "V" formation of flying ducks • Ant colonies • Bees in hives and searching for pollen Humans and machines	Coordination is as fundamental in computation as it is in other parts of nature Coordination in nature - flocking behaviour • Amongst birds, "V" formation of flying ducks • Ant colonies • Bees in hives and searching for pollen Emergent behaviour in complex	Coordination is as fundamental in computation as it is in other parts of nature Coordination in nature - flocking behaviour • Amongst birds, "V" formation of flying ducks • Ant colonies • Bees in hives and searching for pollen Humans and machines

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	- eBay, the online auction company - Customers shopping online by interacting with web site robots to fill in a virtual shopping cart - Second Life and avatars An algorithmic understanding of coordination should lead to successful simulation of natural coordination systems and the delegation of human tasks to machines	systems - when a large number of entities interact, the resulting system can display features and behaviours which are not displayed by the individual constituents. Humans and machines - eBay, the online auction company - Customers shopping online by interacting with web site robots to fill in a virtual shopping cart - Second Life and avatars An algorithmic understanding of coordination should lead to successful simulation of natural coordination systems and the delegation of human tasks to machines	- eBay, the online auction company - Customers shopping online by interacting with web site robots to fill in a virtual shopping cart - Second Life and avatars An algorithmic understanding of coordination should lead to successful simulation of natural coordination systems and the delegation of human tasks to machines
6.1.2. Interaction means that agent behaviours are influenced by information that agents exchange	Online shopping Conway's Game of Life - simple local rules generate features whose dynamic is not explicitly coded in the algorithm. Processing programming language Emergent behaviour in complex systems - StarLogo TNG, Processing programming language	Online shopping Self-organization - a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern Conway's Game of Life - simple local rules generate features whose dynamic is not explicitly coded in the algorithm – see Processing	Online shopping Self-organization - a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern Conway's Game of Life - simple local rules generate features whose dynamic is not explicitly coded in the algorithm – see Processing

Principle	Key Stage 3	Key Stage 4	Key Stage 5
		programming language Emergent behaviour in complex systems - StarLogo TNG, Processing programming language	programming language Emergent behaviour in complex systems - StarLogo TNG, Processing programming language
6.1.3. A protocol is an algorithmic pattern of information exchange among a set of agents	Meaning of term protocol IP addressing Server port numbers for Telnet, HTTP, FTP Use of Telnet for connecting to a web site Use of HTTP via Telnet - Telnet to port 80, issue GET / to retrieve default web page Use of an FTP client for uploading/downloading web pages, etc	Meaning of term protocol End-to-end principle TCP/IP protocol Port numbers Well-known ports - server port numbers Client port numbers Use of Telnet for connecting to a web site Use of HTTP via Telnet - Telnet to port 80, issue GET / to retrieve default web page Use of an FTP client for uploading/downloading web pages, etc	Meaning of term protocol Handshaking protocol in serial and parallel transmission End-to-end principle TCP/IP protocol Port numbers Well-known ports - server port numbers Client port numbers Programmatic use in a programming language such as Delphi: - HTTP protocol (stateless) - Telnet protocol FTP client and server, e.g. use of Cerberus Sending and retrieval of emails: - Use of Telnet to connect to SMTP port - Use of Telnet to connect to POP3 port

Principle	Key Stage 3	Key Stage 4	Key Stage 5
6.1.4. Coordinated interaction means that individual interactions are aligned towards the system's objectives.	Coordinated interaction to achieve a goal Coordination refers to cooperative action Communication refers to the transfer of messages Coordination and communication are not independent	Coordinated interaction to achieve a goal Coordination refers to cooperative action Communication refers to the transfer of messages Coordination and communication are not independent	Coordinated interaction to achieve a goal Multi-threaded programs controlling multiple processors working together to solve a problem Coordination refers to cooperative action Communication refers to the transfer of messages Coordination and communication are not independent
6.1.5. Coordination implies feedback in the interactions so that agents can tell that they are moving toward the system objective. Feedback can be direct (such as one agent acknowledging a request from another) or indirect (such as a merchant adjusting inventory depending on what customers buy).	Importance of feedback	Importance of feedback	Importance of feedback
6.1.6. A game is a framework specifying objectives, players, resources, rules and strategies	A powerful model of coordination systems is the "game" Problem solving - Using lateral thinking to challenge assumptions and establish facts	A powerful model of coordination systems is the "game" Problem solving Understanding the problem (Given initial situation, desired goal	A powerful model of coordination systems is the "game" Problem solving - Understanding the problem (Given initial situation, desired goal

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	- Lateral thinking problem solving activities - Simulation activity based on a game, e.g. SimCity	situation) - Achieving a well-defined problem □ clearly defined initial situation □ clearly defined goal □ clearly defined set of resources □ ownership - Defining boundaries or rules of what and what cannot be done (constraints) - Using lateral thinking to challenge assumptions and establish facts - Planning a solution □ strategies □ what resources to use □ how resources will be used - Lateral thinking problem solving activities	situation) - Achieving a well-defined problem □ clearly defined initial situation □ clearly defined goal □ clearly defined set of resources □ ownership - Defining boundaries or rules of what and what cannot be done (constraints) - Using lateral thinking to challenge assumptions and establish facts/constraints/boundaries - Planning a solution □ strategies □ what resources to use □ how resources will be used - Lateral thinking problem solving activities
6.1.7. A finite game is one that aims to terminate with someone declared a winner	Football match, video game Strategy for never losing at noughts and crosses	Football match, video game Strategy for never losing at noughts and crosses	Football match, video game Strategy for never losing at noughts and crosses
6.1.8. An infinite game is one that aims to continue the play indefinitely	The market economy is an infinite game - players come and go and their objective is to keep the economy going and growing	Virtual worlds, e.g. Second Life, virtual campuses for learning - Open university	Virtual worlds, e.g. Second Life, virtual campuses for learning - Open university

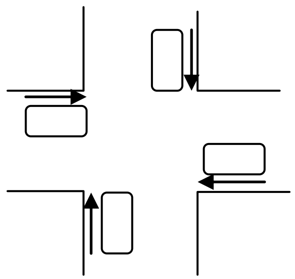
Principle	Key Stage 3	Key Stage 4	Key Stage 5
	The Internet is an infinite game Simulation games - SimCity Virtual worlds, e.g. Second Life, virtual campuses for learning - Open university		
6.2 ESSENTIAL ELEMENTS OF COORDINATION SYSTEMS			
6.2.1. Concurrency and concurrent systems	Concurrency = tasks executed in parallel A concurrent system is a set of tasks, some of which are ordered and the rest concurrent (unordered). Ordered tasks can never execute at the same time; concurrent tasks can.	Concurrency = tasks executed in parallel A concurrent system is a set of tasks, some of which are ordered and the rest concurrent (unordered). Ordered tasks can never execute at the same time; concurrent tasks can.	Concurrency = tasks executed in parallel A concurrent system is a set of tasks, some of which are ordered and the rest concurrent (unordered). Ordered tasks can never execute at the same time; concurrent tasks can.
6.2.2. The game model applies at all three levels of delegation of human tasks to computations. No matter what else they do in the game, the players (humans or their agents) are constantly dealing with five fundamental coordination issues: **Races, exclusive use, arbitration, synchronisation, deadlocks**	Our daily activities have one thing in common -- they require following processes. A process is a sequence of steps that, when followed in order, completes an activity. Processes serve as guides for humans to enable determination of what to do next. Automation of human processes is possible because a computer can be programmed to follow the actions of processes (not all because some	Our daily activities have one thing in common -- they require following processes. A process is a sequence of steps that, when followed in order, completes an activity. Processes serve as guides for humans to enable determination of what to do next. Automation of human processes is possible because a computer can be programmed to follow the actions of processes (not all because some	Our daily activities have one thing in common -- they require following processes. A process is a sequence of steps that, when followed in order, completes an activity. Processes serve as guides for humans to enable determination of what to do next. Automation of human processes is possible because a computer can be programmed to follow the actions of processes (not all

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	processes from human life are beyond a computer's capabilities). There is only a limited amount of space in the world, therefore it is quite possible for the processes that various people are following to conflict with each other. If these conflicts are not solved gracefully, the involved processes may never be completed, so coordination procedures are necessary. Generally, there are five types of coordination that people are concerned with: Races, exclusive use, arbitration, synchronisation, deadlocks.	processes from human life are beyond a computer's capabilities). There is only a limited amount of space in the world, therefore it is quite possible for the processes that various people are following to conflict with each other. If these conflicts are not solved gracefully, the involved processes may never be completed, so coordination procedures are necessary. Generally, there are five types of coordination that people are concerned with: Races, exclusive use, arbitration, synchronisation, deadlocks.	because some processes from human life are beyond a computer's capabilities). There is only a limited amount of space in the world, therefore it is quite possible for the processes that various people are following to conflict with each other. If these conflicts are not solved gracefully, the involved processes may never be completed, so coordination procedures are necessary. Generally, there are five types of coordination that people are concerned with: Races, exclusive use, arbitration, synchronisation, deadlocks.
6.2.3. Races	Relying on computers to buy a train ticket or rent a car can cause problems that wouldn't normally arise with an entirely manual system. For example, assume two people are simultaneously attempting to purchase a train ticket online from Virgin. The system shows one seat available. Does the system sell the seat to both of them? How does the system avoid doing so?	Sometimes several people can be using the same resource for their own purposes, and changes they make to the resource can leave it in a different state depending on who made the last change. The next person who wishes to use the resource can then no longer be guaranteed to find it in a particular state. For example, if three people share the use of a car, and it is agreed that one person will drive to the store to buy groceries and another (at a different time) will fill up	Husband and wife accessing their joint bank account simultaneously from two ATMs Race hazards in logic circuits

Principle	Key Stage 3	Key Stage 4	Key Stage 5
		the gas tank, both while the third person is away, depending on the order in which the first two carry out their chores the third may find either a full tank or a partially spent tank. This problem is known as a race condition, because the outcome varies depending on who "wins" the race.	
6.2.4. Exclusive use	There are many resources in the world which can only be used by one person or group at a time. For example, on a single-lane bridge, only cars going in one direction can pass at a time without causing a collision. Similarly, an airplane seat can be sold only to one person per flight, because multiple people cannot share it. This is known as exclusive use. Usually people can ensure exclusive use through the use of a communication or signalling system. In the former example, traffic signals (and laws about obeying the traffic signals) can bring order to the situation. In the latter example, a system of reservation records can alert an airline employee that a seat has already been sold before another sale is made. Solution: With a mutual excluder, we can safely allow any process to perform an operation on a shared object without interference from another process. Granting exclusive access to a shared	There are many resources in the world which can only be used by one person or group at a time. For example, an airplane seat can be sold only to one person per flight, because multiple people cannot share it. Similarly, on a single-lane bridge, only cars going in one direction can pass at a time without causing a collision. This is known as exclusive use. Usually people can ensure exclusive use through the use of a communication or signalling system. In the former example, a system of reservation records can alert an airline employee that a seat has already been sold before another sale is made. In the latter example, traffic signals (and laws about obeying the traffic signals) can bring order to the situation. Solution: With a mutual excluder, we can safely allow any process to perform an operation on a shared object without interference from	Solution: With a mutual excluder, we can safely allow any process to perform an operation on a shared object without interference from another process. Locking a database record to give exclusive access when writing/updating a record. Locks choose only one process at a time to enter the critical section.

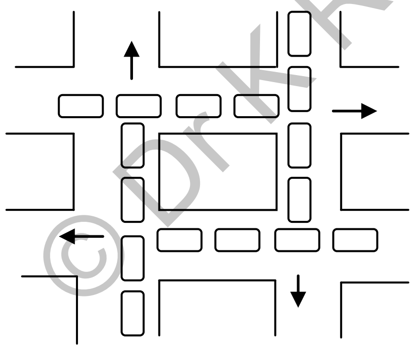
Principle	Key Stage 3	Key Stage 4	Key Stage 5
	printer when printing a job Adventure games Multiplayer networked games	another process. Granting exclusive access to a shared printer when printing a job using a semaphore and queue system.. Multiplayer networked games	
6.2.5. Exclusive Use solution **With a mutual excluder, we can safely allow any process to perform an operation on a shared object without interference from another process**	**Using Semaphores to restrict access to the critical section:** **Scenario:** students have to borrow a book for their class homework of which there is only one copy in the small library of the class. They all have to "share" this book and borrow it one at a time. Without access to a computer database, they need a set of "rules" for checking out the book so that one person can borrow it at a time and inform others that the book is not available. In 1956, E. W. Dijkstra came up with a "great idea" for this problem that used a new variable type, called "semaphore". Using Dijkstra's semaphore idea the students could assign a librarian who would make a list of people trying to borrow the book on first come first served basis (a queue). The librarian could use a flag called a semaphore. When it is 1, it means the book is available. The first person, finding flag =1, checks out the book as	**Using Semaphores to restrict access to the critical section:** **Scenario:** students have to borrow a book for their class homework of which there is only one copy in the small library of the class. They all have to "share" this book and borrow it one at a time. Without access to a computer database, they need a set of "rules" for checking out the book so that one person can borrow it at a time and inform others that the book is not available. In 1956, E. W. Dijkstra came up with a "great idea" for this problem that used a new variable type, called "semaphore". Using Dijkstra's semaphore idea the students could assign a librarian who would make a list of people trying to borrow the book on first come first served basis (a queue). The librarian could use a flag called a semaphore. When it is 1, it means the book is available. The first	**Multiprogramming OS:** Semaphores carry out mutual exclusion and can be used in OSs. Processes in a multiprogramming OS are sent to "sleep" when the shared resource is not available, the same way people sent back home when the book is not available. The operating system gives the process a "wake up" call when the resource becomes available the same way the librarian calls the person to go to the library and check out the book. Incrementing and decrementing the semaphore is also done in a similar manner. A semaphore is an integer variable that, apart from initialization, is accessed only through two standard operations: WAIT and SIGNAL. The WAIT operation checks the value of the semaphore to see whether or not it is greater than 0. If it is, the process can go to the critical section after the value is

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	the librarian's list is empty, and sets the flag to 0. The second (third and so on) person trying to borrow the book, finds out the book is not available (as flag = 0), so the student's name goes into the librarian's list. As soon as the book is checked in, flag is set to 1 again, the librarian calls the first person in the list to borrow the book. The person comes upon the librarian's call, the book is checked out and the flag is set to 0. This process goes on until the queue is empty. BYOB version of Scratch to illustrate use of semaphore.	person, finding flag =1, checks out the book as the librarian's list is empty, and sets the flag to 0. The second (third and so on) person trying to borrow the book, finds out the book is not available (as flag = 0), so the student's name goes into the librarian's list. As soon as the book is checked in, flag is set to 1 again, the librarian calls the first person in the list to borrow the book. The person comes upon the librarian's call, the book is checked out and the flag is set to 0. This process goes on until the queue is empty. BYOB version of Scratch to illustrate use of semaphore.	decremented; if it is not, the process is put to sleep (in a waiting list). The SIGNAL operation increments the value of the semaphore. If one or more process were sleeping on that semaphore, unable to complete an earlier WAIT operation, one of them is chosen and allowed to complete its WAIT. These two operations must be executed indivisibly. Checking the value of semaphore, changing it, and possibly going to sleep (in the WAIT operation) or waking up a process (in SIGNAL operation) is all done as a single atomic action. This way just one process at a time is allowed to go to the critical section. When a process come out of the critical section, the value of semaphore is changed so a process that is asleep is awakened. BYOB version of Scratch to illustrate use of semaphore.
6.2.6. Arbitration	Arbitration arises when a task is required to select only one of two (or more) potentially simultaneous signals, deferring action on the unselected ones without losing them. Whenever there is	By enforcing serial access to a four-way intersection, cars are allowed to travel through in several directions without crashing into one another. Normally, drivers at a four-way stop	Agents are constantly faced with multiple alternative moves, but they can only select one for action. This is problematic if the alternatives are equally attractive. The choice-

Principle	Key Stage 3	Key Stage 4	Key Stage 5
	a need to enforce serial access to a resource (such as with exclusive use), there is always a chance that more than one person will show up at the same time wanting to use it. For example, in a traffic intersection, it is possible for several cars to arrive at once, all seeking to pass through. There has to be some means of determining who gets to go first -- this process is known as arbitration. Sometimes arbitration can be accomplished by mechanisms such as traffic signals, but other times (such as at a four-way stop) a decision must be made on the basis of who arrived first. Unfortunately, this isn't always clear, and making a decision can take an indefinite amount of time when there is no obvious candidate from the start. This situation is known as arbitration failure, and often the only solution is to wait for someone to finally make a decision.	decide who can enter the intersection first on the basis of who arrived first. However, when two drivers arrive simultaneously, convention or the law is invoked which might be right of way is granted to the rightmost driver in the group. But what if drivers arrive from all four directions simultaneously? So what happens then? Often, nothing at all. Each driver is faced with the decision of driving into the intersection first or waiting for one of the other drivers to drive into the intersection first. Picking the former could prove disastrous if any of the other drivers did the same -- picking the latter could force an eternal wait. Neither option is desirable. Eventually the problem is resolved but it is impossible to know how long this will take, and impossible to know who will be the one who moves first, until it happens.	making problem is well recognised by hardware designers, who have to deal with signals that can arrive unpredictably from many sources. It is possible for a standard flip flop (the basic storage element for a bit), on receiving simultaneous signals - clock and data, to enter a metastable state from which there is no definite exit time. It is possible that the flip flop is still metastable at the next clock tick: the circuits that read it behave erratically because they cannot see a definite "0" or "1" from the flip flop. That in turn causes the entire circuit to malfunction or lock up. The flip flop in the processor that tells a processor whether or not an interrupt signal has arrived may enter a metastable state if the signal coincides with the clock signal. The impossibility of guaranteeing a choice within a deadline is called the Choice Uncertainty Principle. A complete solution to the arbitration problem is possible only if the tasks involved can wait an arbitrary length of time for the selection to be made. if there is a deadline there is a probability of arbitration failure - 2 or more signalling tasks, or none may be

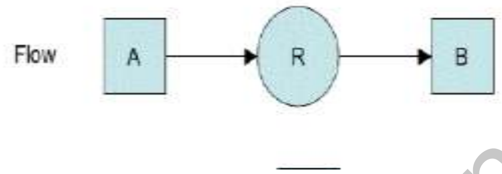
Principle	Key Stage 3	Key Stage 4	Key Stage 5
			selected or unselected signals lost. Choice arbitration can never be eliminated because every level of abstraction at which we prove freedom from synchronisation errors always relies on a lower level at which choice arbitration is solved.
6.2.7. Arbiter **With an arbiter, it is possible to choose one of two simultaneous events, safely.**		Modern computers are subjected to a nearly continuous barrage of incoming signals to process -- keystrokes, mouse movements, and network transmissions, to name a few. All of these signals must be dealt with one at a time in turn. Furthermore, there can be different devices within a single computer attempting to read from or write to the system memory at once, including the main processor (or processors), disk controllers, video cards, and so on. The signals from all of these devices must be serialized to maintain the integrity of the memory. Overall, the number of opportunities for arbitration failure in a computer is much higher than in a simple case such as a building lift controller. Arbitration circuits are therefore required.	One of the first persons to notice that a fundamental principle might be at work in circuits that make decisions was David Wheeler of the University of Cambridge. In the early 1970s, he sought to build a computer whose hardware did not suffer from “hardware freezes” that were common in earlier computers. Wheeler noticed the lockups never occurred when the interrupts were turned off. Interrupt signals were recorded on a flip-flop the processor consulted between instructions: The processor decided either to enter the next instruction cycle or to jump to a dedicated subroutine that responded to the interrupt signal. He suspected that the timing of the interrupt signal’s arrival to that flip-flop occasionally caused it to misbehave and hang the computer. **Arbiter Circuit** Unambiguous choices between

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			near-simultaneous signals must be made in many parts of computing systems, not just at interrupt flip-flops. Examples: • Two processors request access to the same memory bank (dual-ported random-access memory (RAM)). • Two transactions request a lock on the same record of a database. • Two external events arrive at an object at the same time. • Two computers try to broadcast on an Ethernet at the same time. • Two packets arrive together at the network card
6.2.8. Synchronisation **With a synchronizer, one process can be forced to stop and wait for a signal from another.**	Signalling systems to prevent trains from colliding on a section of track or cars colliding at crossroads. Whether we use clocks or signals, our real purpose is to coordinate actions with other people and with machines. The cost of a failed synchronization may be an inconvenience, a loss of business, or an accident. Synchronization -- meaning to perform actions together -- is a fundamental aspect of coordination.		Synchronisation is a requirement that a task cannot proceed past a point until another task signals that it has passed a corresponding checkpoint. Semaphore with Wait and Signal operations is a model for synchronisation.

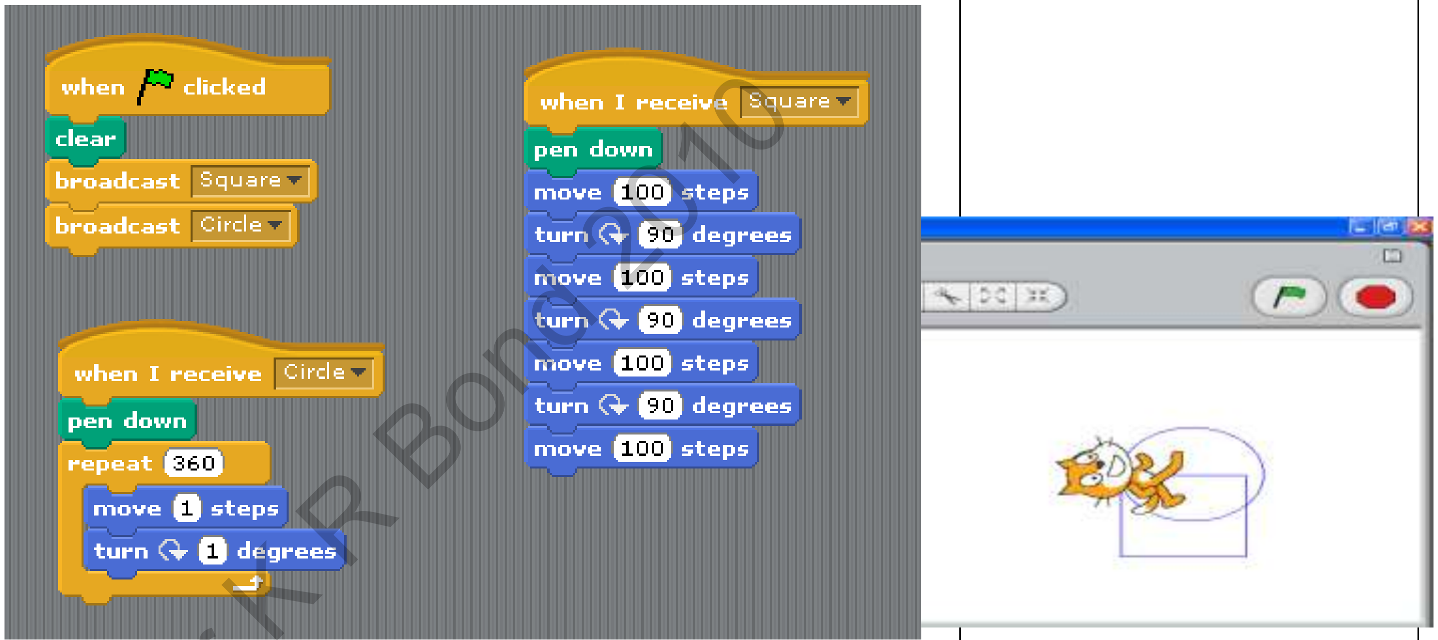
Principle	Key Stage 3	Key Stage 4	Key Stage 5
6.2.9. Deadlocks	Deadlock is a condition in which a set of processes are all stopped and waiting for a signal from another member of the set. Deadlocks occur in human systems quite frequently, e.g. traffic gridlock. In a multi-agent systems it is possible under the right circumstances for a set of agents to become entangled in a circular wait. That means that each one is stopped waiting for another in the set to send a signal. Since everyone is waiting, no one can actually send a signal. Deadlock arises because of an insufficiency of resources. Each car does not have enough road space to proceed. 	Two agents request two resources in different orders: A asks for (R1, R2) and B for (R2, R1). If both initiate their requests at the same time, they can enter a state where A holds R1 and B holds R2. When A then asks for R2, it must wait since B already holds R2. When B asks for R1, it must wait since A already holds R1. Now both A and B are waiting for the other to release a resource. They are both stuck. Neither can do anything. This is a deadlock situation. One solution is to require all agents to request resources in some preset order. Another is if all tasks obtain all resources they need before commencing execution. Two agents can also get into deadlock because of an insufficiency of resources they have been authorised to use. Total amount of resource = 1000, A requests and gets 600, B requests and gets 400, then A requests another 200 and B another 300. Both A and B will now have to wait.	Multiprogramming OS - deadlock avoidance and detection

Principle	Key Stage 3	Key Stage 4	Key Stage 5
6.3 THE PROTOCOLS OF COORDINATION SYSTEMS MANAGE DEPENDENCIES OF FLOW, SHARING AND FIT AMONG ACTIVITIES			
6.3.1. Coordination is required if two or more activities depend on each other in some way. The nature of the dependencies follows from the game. The three basic types of dependency are flow, sharing and fit.	Because there is only a limited amount of space in the world, it is quite possible for the processes that various people are following to conflict with each other. If these conflicts are not solved gracefully, the involved processes may never be completed, so coordination procedures are necessary. We coordinate many actions in life with the help of a clock. Times are set for such events as meetings, classes, train and airplane departures - history of keeping time accurately, time coordination, time zones, atomic clocks, timing signal distribution across communication networks, accuracy of GPS systems and special relativity. Signalling systems are set up to prevent trains from colliding on a section of track or cars colliding at cross roads. Whether we use clocks or signals, the real purpose is to *coordinate actions* with other people and with machines. Coordination problems can be solved, in time, by people who encounter them in the course of following processes, largely by relying on past experience and reason. However, computers and other automated machinery have neither past	Generally, there are five types of coordination that people are concerned with: Races, exclusive use, arbitration, synchronisation, deadlocks Each of these five coordination problems can be solved, in time, by people who encounter them in the course of following processes, largely by relying on past experience and reason. However, computers and other automated machinery have neither past experience nor reason to rely upon for answers. If they are to coordinate processes successfully, the procedures for coordination must be just as automated as the processes themselves. It turns out that foolproof, automated coordination mechanisms can be very difficult to implement, because things that the human brain can reduce to "obvious" are not necessarily so in actuality. Given the vast number of computer programs that run concurrently in the world today, crafting reliable coordination schemes for all of them	Generally, there are five types of coordination that people are concerned with: Races, exclusive use, arbitration, synchronisation, deadlocks Each of these five coordination problems can be solved, in time, by people who encounter them in the course of following processes, largely by relying on past experience and reason. However, computers and other automated machinery have neither past experience nor reason to rely upon for answers. If they are to coordinate processes successfully, the procedures for coordination must be just as automated as the processes themselves. It turns out that foolproof, automated coordination mechanisms can be very difficult to implement, because things that the human brain can reduce to "obvious" are not necessarily so in actuality. Given the vast number of computer programs that run concurrently in the world today, crafting reliable

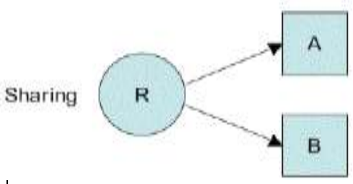
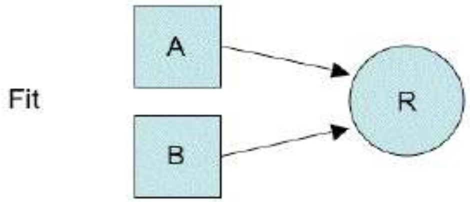
Principle	Key Stage 3	Key Stage 4	Key Stage 5
	experience nor reason to rely upon for answers. If they are to coordinate processes successfully, the procedures for coordination must be just as automated as the processes themselves. It turns out that foolproof, automated coordination mechanisms can be very difficult to implement, because things that the human brain can reduce to "obvious" are not necessarily so in actuality. A dependency exists between activities A and B when the completion of one (say B) depends in some way on the other (A).E.g. - Event A must precede event B - B needs information from A before acting - A and B both need the same processor - A and B produce parts that are combined into a single assembly - A's inputs to B must be in formats recognised by B Dependency patterns Flow - flowcharting Sharing - shared resource, B may be forced to wait until A releases shared resource	would be a nearly impossible task. Yet, we cannot risk the corruption of valuable data that would occur by ignoring these five problems.	coordination schemes for all of them would be a nearly impossible task. Yet, we cannot risk the corruption of valuable data that would occur by ignoring these five problems.

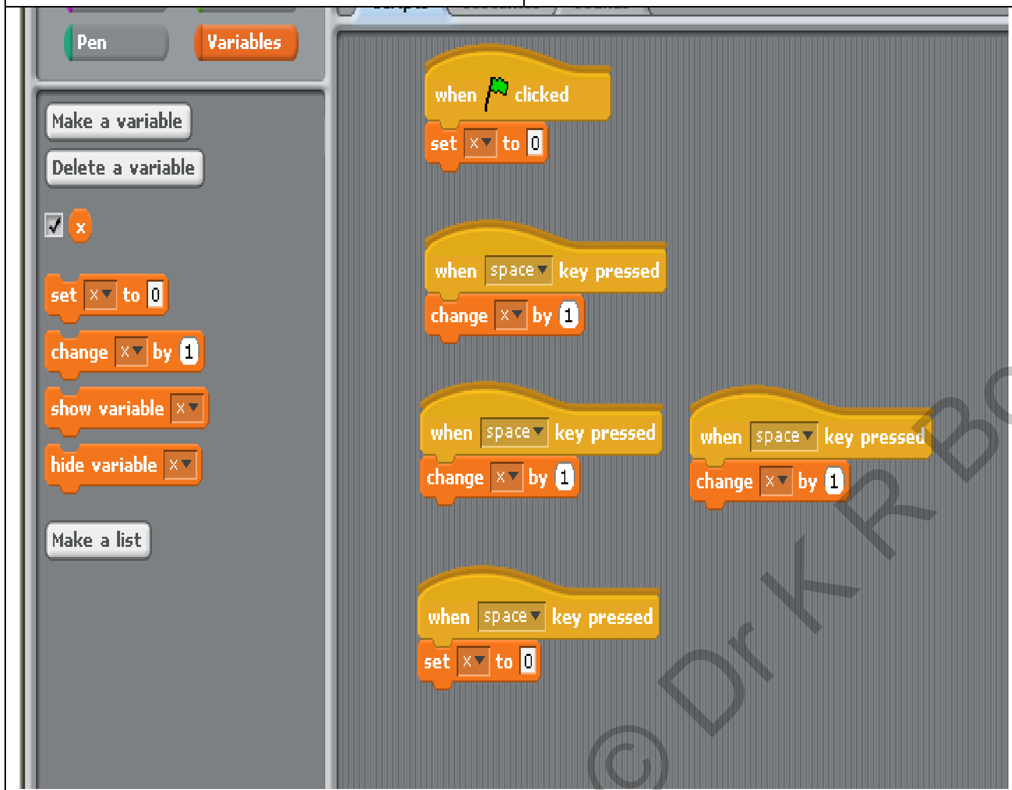
Principle	Key Stage 3	Key Stage 4	Key Stage 5
	Fit - A and B update a single resource		
6.3.2. A flow dependency exists when one activity produces a resource/signal/message that is required/used by another activity. Message sending, signalling and flowcharting are examples	Flowcharts 		Flowcharts A's input to B must be in formats recognised by B: e.g. compatible file types, XML

Principle	Key Stage 3	Key Stage 4	Key Stage 5



Flowcharts

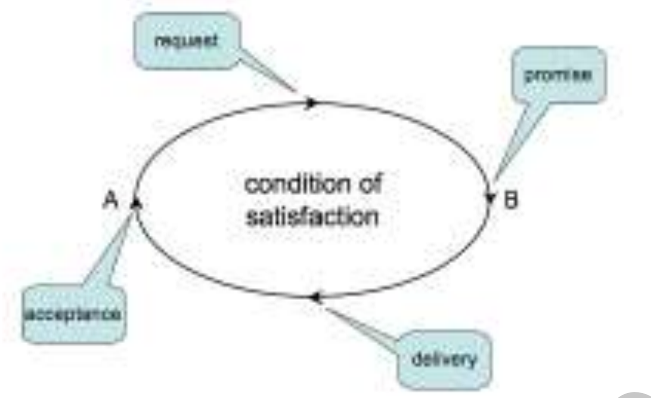
Principle	Key Stage 3	Key Stage 4	Key Stage 5
6.3.3. A sharing dependency – **multiple activities all using the same (usually limited) resource** 	Processor sharing Accessing a shared database Sharing a network printer Sharing a networked hard drive in a network OS Sharing main memory in a multitasking OS	Processor sharing Accessing a shared database Sharing a network printer Sharing a networked hard drive in a network OS Sharing main memory in a multitasking OS	Processor sharing Accessing a shared database Sharing a network printer Sharing a networked hard drive in a network OS Sharing main memory in a multitasking OS
6.3.4. Fit dependency – **Multiple activities collectively producing, contributing to, or updating a single resource** 	What is value of x after program is started and space bar is pressed (see row below)? The answer is unpredictable. Multi-user games such as Halo and Quake	Modular development of a software system Different travel agency offices booking seats on the same flight - lost update problem Multi-user games such as Halo and Quake	Modular development of a software system Different travel agency offices booking seats on the same flight - lost update problem

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
6.3.5. A coordination process is a mechanism implementing these dependencies. Many alternatives are available for each dependency.	FIFO queuing - printer spooling, Web server sharing - requests queued XML	FIFO queuing - printer spooling, Web server sharing - requests queued	FIFO queuing - printer spooling, Web server sharing - requests queued Round Robin scheduling of processor Mutual exclusion locks
6.4 COORDINATION TASKS CAN BE DELEGATED TO COMPUTATIONAL PROCESSES			
6.4.1. Humans delegate tasks to agents by designing computational processes to perform the tasks. **There are three categories of coordination systems according to the amount of task delegation:** **1. Human-human with computer assistance: all interaction is between humans, but computational processes track their joint progress states to assist them to complete tasks. (Known as Computer Supported Cooperative Work, CSCW).**	1.Google groups • Interactions of Alice and Bob tracked by a computation whose records help them to monitor each task's progress towards completion. 2. Flight simulation • System presents pilot with scenes that might appear through cockpit windows. • Moving the controls causes the scene to move in exactly the way it would appear as the airplane responds. • Object is for the pilot to learn to respond to situations in the right ways prior to actually encountering them in real flight.	1.Google groups • Interactions of Alice and Bob tracked by a computation whose records help them to monitor each task's progress towards completion. 2. Flight simulation • System presents pilot with scenes that might appear through cockpit windows. • Moving the controls causes the scene to move in exactly the way it would appear as the airplane responds. • Object is for the pilot to learn to respond to situations in the right ways prior to actually encountering them	1.Google groups • Interactions of Alice and Bob tracked by a computation whose records help them to monitor each task's progress towards completion. 2. Flight simulation • System presents pilot with scenes that might appear through cockpit windows. • Moving the controls causes the scene to move in exactly the way it would appear as the airplane responds. • Object is for the pilot to learn to respond to situations in the right ways

Principle	Key Stage 3	Key Stage 4	Key Stage 5
2. Human-computer: the performer role is delegated to a computational system. Humans interact with the system through an interaction language and interaction interface. (Known as Human Computer Interaction, HCI). **3. Computer-computer: all requester and performer roles are delegated to computational processes. All interactions are carried out automatically between machines. (Known as concurrency controls, CC).**	• How should displays be designed so that complete situational information is conveyed? 3. Monitoring of network ports for incoming packets + routing to subsystem that processes the packets. • incoming web page packet routed to TCP system and to TCP/IP application, web browser.	in real flight. • How should displays be designed so that complete situational information is conveyed? 3. Monitoring of network ports for incoming packets + routing to subsystem that processes the packets. • incoming web page/FTP packet routed to TCP system and to TCP/IP corresponding application.	prior to actually encountering them in real flight. • How should displays be designed so that complete situational information is conveyed? 3. Monitoring of network ports for incoming packets + routing to subsystem that processes the packets. • OS provision of automatic context switching and round robin scheduling to create illusion that many computational processes (threads) exist in simultaneous execution. • incoming packets routed to TCP system and to TCP/IP application.

Principle	Key Stage 3	Key Stage 4	Key Stage 5
6.5 ACTION LOOP IS THE FOUNDATIONAL ELEMENT OF ALL COORDINATION PROTOCOLS			
6.5.1 Most individual human interactions are modelled by an action loop in which a performer delivers a condition satisfying a customer. The loop has a requestor (A) and performer (B) and four time segments culminating in request, promise, delivery and acceptance. Before the loop starts, the condition is not true; when it completes, the condition is true. **CSCW - Computer Supported Cooperative Work** **HCI - Human Computer Interaction** **CC - Concurrency Control** **-**	The most fundamental human coordination pattern between two parties A and B is the action loop: A: I request B: I accept B: I deliver A: I am satisfied Action loops are pervasive in natural systems - e.g. coordination dances of ants and bees are built of action loops - e.g. animal social systems exhibit hierarchies in which more dominant individuals consolidate their power by initiating more action loops Request-acknowledge loop between two hardware components - to prevent the initiating component from sending another request until the other component is ready Simple human-human analogy Online purchasing	The most fundamental human coordination pattern between two parties A and B is the action loop: A: I request B: I accept B: I deliver A: I am satisfied Action loops are pervasive in natural systems - e.g. coordination dances of ants and bees are built of action loops - e.g. animal social systems exhibit hierarchies in which more dominant individuals consolidate their power by initiating more action loops Request-acknowledge loop between two hardware components - to prevent the initiating component from sending another request until the other component is ready Request - Response model of client-server interaction - HTTP application protocol in TCP/IP □ Request message for a web page - GET / □ Response message - web page	The most fundamental human coordination pattern between two parties A and B is the action loop: A: I request B: I accept B: I deliver A: I am satisfied Action loops are pervasive in natural systems - e.g. coordination dances of ants and bees are built of action loops - e.g. animal social systems exhibit hierarchies in which more dominant individuals consolidate their power by initiating more action loops Request-acknowledge loop between two hardware components - to prevent the initiating component from sending another request until the other component is ready - Serial and parallel communication □ RS232 □ USB ○ Arduino microcontroller boards □ Bluetooth ○ WII remote controller □ Serial ATA □ Parallel ATA □ Centronics parallel interface

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			Request - Response model of client-server interaction - Application protocols in TCP/IP □ Request message □ Response message Server-side - Request object - Response object
			

Principle	Key Stage 3	Key Stage 4	Key Stage 5

A	B	
human	human	CSCW
human	computer	HCI
computer	computer	CC

Principle	Key Stage 3	Key Stage 4	Key Stage 5

Learning outcomes for Key Stages 3 to 5 Computing. Strand: AUTOMATION

Learners should know and understand……

7. Automation **These principles concern finding efficient computational ways to perform human tasks. Tasks can be physical, such as running an assembly line, driving a car, controlling airplane surfaces; or mental, such as doing arithmetic, playing chess, and planning schedules.**			
7.1 Physical automation maps hard computational tasks to physical systems that perform them acceptably well.			
7.1.1. The class of very hard computational tasks includes most of the problems people want to solve for business, science, and engineering; **for example, figuring route capacities in a transportation network or numerically simulating an aircraft in flight. These problems have all been proved to be NP-hard or worse.**			NP-hard problems

Principle	Key Stage 3	Key Stage 4	Key Stage 5
7.1.2. There is tremendous motivation to find approximate or heuristic ways to solve them, not perfectly but acceptably well. Very hard tasks are mapped to simpler systems that do good-enough jobs. This is a form of automation because what was previously beyond reach can now be done well by a machine. It is not unusual for a heuristic system that does consistently well with a (formerly) computationally hard task to be celebrated as a technology breakthrough.	Robotics and Control – Sensing – Measuring – Controlling E.g. Lego Mindstorms, Soda Race, Jeroo, Kara, Processing programming language and ARDUINO microcontroller boards Object identification and object identification filing E.g. Apple iphone Face recognition and face recognition filing Image recognition and image recognition filtering E.g. Picassa photo album software from Google Collaborative wiki-like learning systems, e.g. www.phrasedetectives.org	Robotics and Control – Sensing – Measuring – Controlling E.g. Lego Mindstorms, Soda Race, Jeroo, Kara Object identification and object identification filing E.g. Apple iphone Face recognition and face recognition filing Image recognition and image recognition filtering E.g. Picassa photo album software from Google Collaborative wiki-like learning systems, e.g. www.phrasedetectives.org	Automated planning and scheduling – intelligent agents – autonomous robots – unmanned vehicles The solutions are complex, unknown and have to be discovered and optimised in multidimensional space. Heuristics Pattern recognition - "the act of taking in raw data and taking an action based on the category of the data" Pattern recognition aims to classify data based on either a priori knowledge or on statistical information extracted from the patterns. A complete pattern recognition system consists of a sensor hat gathers the observations to be classified or described; a feature extraction mechanism that computes numeric or symbolic information from the observations; and a classification or description scheme that does the actual job of classifying or describing observations, relying on the extracted features. The classification or description

Principle	Key Stage 3	Key Stage 4	Key Stage 5
			scheme is usually based on the availability of a set of patterns that have already been classified or described. This set of patterns is termed the training set and the resulting learning strategy is characterised as supervised learning. For example, Learning can also be unsupervised in the sense that the system is not given an *a priori* labelling of patterns, instead it establishes the classes itself based on the statistical regularities of the patterns.
7.1.3. The artificial intelligence branch of the computing field has developed and studied search processes that reduce very hard tasks to workable ones. For example, a genetic algorithm can generate a set of candidate solutions to the problem and then transform them through several generations of cross-matches and mutations until it evolves a good enough solution.			Genetic algorithms
7.1.4. People in the AI field work with two different hypotheses about human tasks:	Primitive brain • infant brain and primitive reflex	"Weak" AI just claims the digital computer is a useful tool for studying intelligence and developing useful technology.	"Weak" AI just claims the digital computer is a useful tool for studying intelligence and developing useful technology.

Principle	Key Stage 3	Key Stage 4	Key Stage 5

• **The strong AI hypothesis holds that human tasks are ultimately computational; they are a subset of hard computational tasks. Strong AI seeks to discover the computational methods and automate them.**
 	
• **The weak AI hypothesis holds that human tasks may not be computational but that we can nonetheless find good-enough computational equivalents. Weak AI seeks to construct computational systems that mimic human tasks acceptably well.**
	behaviour
	Cerebral cortex
	• Development of neural pathways in cerebral cortex
	• Suppression of primitive brain connections
	Vision
	Visual computation divided into an unconscious parallel stage and a conscious serial stage. Unconscious part of the visual field sprinkled with thousands of little processors. Each detects a colour or a simple shape like a curve whenever it appears at the processor's location. The output of another set looks like: straight straight curved straight straight straight and so on. Superimposed on top of these processors is a layer of odd-man-out detectors.
	• Unconscious processing
	  ◦ Shape detecting processors
	  ◦ Odd man out detector
	• Conscious processing
	  ◦ Optical illusions
		A running AI program is at most a simulation of a cognitive process but is not itself a cognitive process. Analogously, a meteorological computer simulation of a hurricane is not a hurricane.
		"Strong" AI claims that a digital computer can in principle be programmed to actually BE a mind, to be intelligent, to understand, perceive, have beliefs, and exhibit other cognitive states normally ascribed to human beings.
			A running AI program is at most a simulation of a cognitive process but is not itself a cognitive process. Analogously, a meteorological computer simulation of a hurricane is not a hurricane.
			"Strong" AI claims that a digital computer can in principle be programmed to actually BE a mind, to be intelligent, to understand, perceive, have beliefs, and exhibit other cognitive states normally ascribed to human beings.
			
			Functionalism vs Behaviourism vs Identity Theory
			
			Searle's Chinese Room
			
			Turing Test

Mapping Principles to Resources

Principle	Key Stage 3	Key Stage 4	Key Stage 5

Principle	Key Stage 3	Key Stage 4	Key Stage 5
1.1.1	http://csunplugged.org/binary-numbers - Activity 1 "Count the Dots - Binary Numbers" CSInside "Painting By Numbers" http://www.info-study.net/unplugged/activity2-j.html - pixel representation flash animation	http://greenroom.greenfoot.org/resources/4 - fax machine greenfoot exercise Monochrome bitmap in Excel http://csunplugged.org/image-representation - Activity 2 "Colour By Numbers"	Nelson Thornes AS level and A2 level Computing textbooks http://www.cs4fn.org/pixels/pixels.html - Pixel Puzzle http://www.ece.uc.edu/mc2/browse.php?level0=&level1=&level2=&level3=&id=2 - Healthcare imaging CSInside - "In the Picture"
1.1.2			
1.1.3			
1.1.4		http://csunplugged.org/information-theory - Activity 5 "Twenty Guesses - Information Theory"	
1.1.5			
1.1.6			
1.1.7			
1.1.8			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
1.1.9	http://csunplugged.org/sorting-algorithms - Activity 7 – Sorting algorithms http://csunplugged.org/searching-algorithms - Activity 6 - Searching Algorithms		
1.1.10			
1.1.11			
1.1.12			
1.1.13			
1.1.14			
1.1.15			
1.1.16			
1.2.1	http://csunplugged.org/text-compression - Activity 3 - "Text Compression"	CSInside - "Zipping It Up" http://www.cs4fn.org/internet/crushed.php - "Little Data: Compressing Vicki Pollard"	

Principle	Key Stage 3	Key Stage 4	Key Stage 5
1.2.2			
1.2.3			
1.2.4			http://www.cs4fn.org/graphics/casinoroyalefractal.php - "Fractal Casino Royale"
1.2.5	http://www.cs4fn.org/mathemagic/sonic.html - The Magic of MP3 http://www.cs4fn.org/films/jpegit.php - "Picture This? JPEG IT!"		
1.2.6			
1.3.1			Nelson Thornes A2 level Computing textbook
1.3.2			
1.4.1			CSInside - "Finding a Needle in a Haystack"
2.1.1			Turing machine - http://aturingmachine.com/index.php Nelson Thornes A2 Computing textbook

Principle	Key Stage 3	Key Stage 4	Key Stage 5
2.1.2	CSInside - "Algorithm development" Algorithms worksheet		
2.1.3	Programming in Scratch, Gamemaker or Alice	Programming in an event-driven programming language	Programming in an event-driven programming language
2.1.4			
2.2.1	CSInside - "Algorithm Development" Raptor	http://www.cs4fn.org/algorithms/swappuzzle/ - Swap Puzzle Raptor	Nelson Thornes AS and A2 level textbook
2.2.2		http://csunplugged.org/finite-state-automata - "Finite State Automata" Kara - http://www.swisseduc.ch/compscience/karatojava/kara/	Nelson Thornes AS and A2 level textbook Kara - http://www.swisseduc.ch/compscience/karatojava/kara/
2.2.3			Nelson Thornes AS level textbook
2.2.4			
2.2.5		Role of variables - http://cs.joensuu.fi/~saja/var_roles/	Role of variables - http://cs.joensuu.fi/~saja/var_roles/

Principle	Key Stage 3	Key Stage 4	Key Stage 5
2.2.6	Scratch programming		
2.2.7			
2.2.8			Nelson Thornes A2 level textbook
2.2.9			Nelson Thornes A2 level textbook
2.2.10			
2.2.11			
2.2.12			
2.3.1			
2.3.2			
2.3.3			
2.3.4			
2.3.5			
2.4.1			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
2.4.2			
2.4.3			
2.4.4			
2.4.5			
2.4.6			
2.4.7			
2.4.8			
2.4.9			http://csunplugged.org/graph-colouring - Activity 13 - Graph Colouring
2.4.10			http://csunplugged.org/dominating-sets - Activity - Dominating Sets
3.1.1			
3.1.2			
3.1.3			

Principle	**Key Stage 3**	**Key Stage 4**	**Key Stage 5**
3.1.4			
3.1.5			
3.2.1			
3.2.2			
3.2.3			
3.2.4			
3.2.5			
3.2.6			
3.2.7			
3.3.1			
3.3.2			
3.4.1			
3.4.2			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.4.3			
3.5.1			
3.5.2	http://csunplugged.org/cryptographic-protocols - Activity 17 - Cryptographic Protocols		
3.5.3			
3.5.4			
3.5.5			
3.5.6			
3.5.7			
3.6.1			
3.6.2			
3.6.3			
3.6.4			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.6.5			
3.6.6			
3.7.1			
3.7.2			
3.7.3			
3.7.4			
3.7.5			
3.7.6			
3.7.7			
3.7.8			
3.7.9			
3.7.10			
3.7.11			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.7.12			
3.7.13			
3.7.14			
3.8.1			
3.8.2			
3.8.3			
3.8.4			
3.9.1			
3.9.2			
3.9.3			
3.9.4			
3.9.5			
3.9.6			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
3.10.1			
3.10.2			
3.11.1			
3.11.2			
3.11.3			
3.11.4			
3.11.5			
3.11.6			
3.11.7			
3.11.8			
3.12.1			
3.12.2			
3.12.3			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
4.1.1	http://education.mit.edu/drupal/starlogo-tng/learn - StarLogo TNG tutorials	http://education.mit.edu/drupal/starlogo-tng/learn - StarLogo TNG tutorials	http://education.mit.edu/drupal/starlogo-tng/learn - StarLogo TNG tutorials
4.1.2			
5.1.1			
5.1.2			
5.1.3			
5.1.4			
5.2.1			
5.2.2			
5.2.3			
5.2.4			
5.3.1			
5.3.2			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
5.3.3			
5.3.4			
5.3.5			
5.3.6			
5.3.7			
5.4.1			
5.4.2			
5.4.3			
5.4.4			
5.5.1			
6.1.1	http://education.mit.edu/drupal/starlogo-tng/learn - StarLogo TNG tutorials	http://education.mit.edu/drupal/starlogo-tng/learn - StarLogo TNG tutorials	http://education.mit.edu/drupal/starlogo-tng/learn - StarLogo TNG tutorials
6.1.2	http://education.mit.edu/drupal/starlogo-tng/learn - StarLogo TNG tutorials	http://education.mit.edu/drupal/starlogo-tng/learn - StarLogo TNG tutorials	http://education.mit.edu/drupal/starlogo-tng/learn - StarLogo TNG tutorials

Principle	Key Stage 3	Key Stage 4	Key Stage 5
6.1.3			
6.1.4			
6.1.5			
6.1.6			
6.1.7			
6.1.8			
6.2.1			
6.2.2			
6.2.3			
6.2.4			
6.2.5			
6.2.6			
6.2.7			

Principle	Key Stage 3	Key Stage 4	Key Stage 5
6.2.8			
6.2.9	http://csunplugged.org/routing-and-deadlock - Activity 10 - The Orange Game—*Routing and Deadlock in Networks*		
6.3.1			
6.3.2			
6.3.3			
6.3.4			
6.3.5			
6.4.1			
6.5.1			
7.1.1			
7.1.2			
7.1.3			
7.1.4			

Themes

The following are some possible themes that could form the basis of a teaching programme:

1. Robotics
2. Animation, and reactive programming
3. Uses of bit patterns
4. Simulation
5. Secret messages and cryptography
6. What computers can and cannot do (complexity and computability)
7. Artificial intelligence
8. Games (video games yes, but also Nim, chequers, sudoku, mastermind, etc where the computer is an active player)
9. Fun with graphs (apply graph theory in different settings)
10. Emergent behaviour (life, chaos, ants, cellular automata etc, neural networks)
11. Coordination

Programmes of Study

Key Stage 2

Unit	Project Title	Knowledge & Skills	Breadth of study	Theme
2.1				Animation & Reactive Programming
2.2				Secret Messages & Encryption
2.3				Uses of bit patterns
2.4				Robotics & Control

Key Stage 3

Unit	Project Title	Knowledge & Skills	Breadth of study	Theme
3.1				Animation & Reactive Programming
3.2				Secret Messages & Encryption
3.3				Robotics

3.4				Simulation
3.5				Uses of bit patterns
3.6				Coordination

Key Stage 4

Unit	Project Title	Knowledge & Skills	Breadth of study	Theme
4.1				Games Programming
4.2				Emergent behaviour
4.3				Robotics
4.4				Simulation
4.5				Information Retrieval
4.6				What computers can and cannot do
				Artificial Intelligence

RESOURCES

Schemes of Work

Clive Hirst of Merchant Taylors' School, Northwood has incorporated many of the ideas contained within this BOK into schemes of work for Key Stages 3 and 5 - http://www.asiplease.net/computing/

Emma Wright, Harveys' Grammar School, Folkstone, Kent has devised a KS3 Year 9(?) scheme of work based on the principles model – see next page.

Term 1 Computation

Week	Lesson Ref		Hour 1	Principle	Lesson Ref	Hour 2	Principle
05-Sep	T1.W1.L1	Computation	1.1.1	A representation is a pattern of symbols that conveys information	T1.W1.L2	1.1.5	Every representation is embodied into physical phenomena
12-Sep	T1.W2.L1		1.1.4 and 1.1.7	Meaning is discerned and acted upon by observers reading pattern	T1.W2.L2	1.1.8	Interpretation of a representation depends on the means by which observers interact with the representation
19-Sep	T1.W3.L1		1.1.11	Rules describe the allowable patterns of carrier configurations	T1.W3.L2	1.1.12	Representations are finite
26-Sep	T1.W4.L1		1.1.13	Representations can represent the infinite	T1.W4.L2	1.1.14	Representations are equivalent if they represent the same information
03-Oct	T1.W5.L1		1.1.15	Linear and non-linear representations	T1.W5.L2	?	Computations in the real world can involve finite and continuous representations
10-Oct	T1.W6.L1		1.1.2 and 1.1.6	Symbols can be encoded with patterns of bits. Continuous representations also because any value of the representation function can be approximated by a binary number.	T1.W6.L2	1.1.19	An algorithm is a representation of a method to accomplish a task or process. An algorithm transforms input data to output data. Input data are states of an input representation which can be thought of as a set of symbols that represent values. Output data are likewise the states of an output representation. A program is an algorithm plus its data.
17-Oct	T1.W7.L1		Examination		T1.W7.L2	Evaluation	

Term 2 Computation

Week		Lesson Ref	Hour 1	Principle	Lesson Ref	Hour 2	Principle
31-Oct	Computation	T2.W1.L1	1.1.10	A compiler translates a program to machine code, the low-level bit patterns that drive a machine. An algorithm can be regarded as a logical machine because of the equivalence between a high-level language program and its machine code representation as produced by a compiler.	T2.W1.L2	1.1.16	Achieving equivalence of stages of development of a program: Requirements Specifications Source language Compiled code Compiled code = original requirements

07-Nov		**T2.W2.L1**	2.2.1	Computations are not just manmade products of manmade computers. Computing machines made by man are just one way of realising computations	**T2.W2.L2**	2.2.2	A computation is a sequence of states of a data representation caused by an algorithm. States represent values. Successive representations are controlled by logic rules embodied in operators. An operator causes a specific, precise change in total state.
14-Nov		**T2.W3.L1**	2.2.3	A computing machine is a physical system or process for holding a representation and acting upon it by an algorithm. The "machine" can be based on digital electronics or it can be something else, e.g. DNA Machine instructions are machine-level operators	**T2.W3.L2**	2.2.4	An operator causes a specific, precise change in total state. Machine instructions are machine-level operators. Most operators alter a confined, finite portion of a state. Some operators can alter the entire state Given an initial state, an algorithm specifies how operators from a finite set are applied to produce a final state: in what order and how many times.
21-Nov		**T2.W4.L1**	2.2.5	An algorithm's total state is a record of the values of all its input, output, internal (private) variables and external variables. These variables are specified in the algorithm's data representation.	**T2.W4.L2**	2.2.6	Organise programs so that their dynamic computations mirror their textual structure in order to make algorithms more understandable and to reduce errors. Control structures expressed with just four basic forms: Procedure call Sequence Selection Iteration
28-Nov		**T2.W5.L1**	2.2.7	Data-oriented computational forms Computations driven by interactions with the external environment of a computation and by particular data that the computation receives at each interaction point	**T2.W5.L2**	2.2.8	If an algorithm's data representation has an infinite number of states, the algorithm can generate a potentially infinite number of computations. Thus an algorithm is a highly compressed representation of a very large, potentially infinite, space of computations. This is why algorithms are difficult to understand and prove correct
05-Dec		**T2.W6.L1**	2.2.9	Computation is unavoidable: The only general method of approaching the question of whether computations halt or produce useful results is to run them and see what happens. This conclusion gave birth to Computer Science.	**T2.W6.L2**	**Peer evaluation & Assessment / Presentation**	
12-Dec		**T2.W7.L1**	**Examination**		**T2.W7.L2**	**Evaluation**	

Term 5 DESIGN

Week		Lesson	Hour 1	Principle	Lesson	Hour 2	Principle
16-Apr	Computation	T5.W1.L1	5.1.1	Modularity is a process of dividing a large system into a hierarchy of aggregates (modules) that interact across precisely defined interfaces.	T5.W1.L2	5.1.2	Abstraction means to define a simplified version of something and to state the operations (functions) that apply to it. By bringing out the essence and suppressing detail, an abstraction offers a simple set of operations that apply to all the cases. In a hierarchy, an abstraction corresponds to an aggregate; forming a hierarchy is a process of abstraction. Abstraction is one of the most fundamental powers of the human brain.
23-Apr		T5.W2.L1	5.1.3	Information hiding means to hide the details of an implementation so that users do not see them. It protects against errors caused by changes in the details that do not concern users. It is a policy that supports abstraction by preventing users of the abstraction from gaining access to the suppressed details behind the abstraction. In a hierarchy, it is a decision to hide the component structure of an aggregate, allowing that structure to be rearranged without changing the behavior of the aggregate. A software module implements a software function by hiding internal details behind a simple interface.	T5.W2.L2	5.1.4	Decomposition means to subdivide a large problem into components that can be designed separately and then assembled into the full system. In a hierarchy, identifying the components of an aggregate is an act of decomposition. A module is an abstraction of the components that compose it.
30-Apr		T5.W3.L1	5.3.1	Design means two things: architecture and process. Architecture is a division of a system into components, their interactions and their layout. Process is the steps producing an architecture.	T5.W3.L2	5.3.2	Design process is adopted from engineering 1. Requirements 2. Specifications 3. Prototype 4. Testing
07-May		T5.W4.L1	5.3.3	Four main criteria for good design: 1. Correctness 2. Speed 3. Tolerance 4. Fit	T5.W4.L2	5.3.4	Correctness means that the software system provably meets precise specifications. Correctness is challenging because of the difficulty of getting precise specifications for complex systems and the computational intractability of formal proofs for large systems

14-May		T5.W5.L1	5.3.5	Speed means that the system completes tasks within acceptable time limits.	T5.W5.L2	5.3.6	Fault tolerance means that the software and host systems can continue to function despite small errors and will refuse to function in the case of large errors.
21-May		T5.W6.L1	5.3.7	Fitness means that the dynamic behaviour of the system aligns with the environment of its use.	T5.W6.L2	Examination	

Term 6

COORDINATION

Week		Lesson	Hour 1	Principle	Lesson	Hour 2	Principle
04-Jun	Coordination	T6.W1.L1	6.2.1	Concurrency and concurrent systems	T6.W1.L2	6.2.2	The game model applies at all three levels of delegation of human tasks to computations. No matter what else they do in the game, the players (humans or their agents) are constantly dealing with five fundamental coordination issues: Races, exclusive use, arbitration, synchronisation, deadlocks
11-Jun		T6.W2.L1	6.2.3	Races	T6.W2.L2	6.2.4	Exclusive use
18-Jun		T6.W3.L1	6.2.4	Exclusive use solution 1	T6.W3.L2	6.2.5	Exclusive Use 2 With a mutual excluder, we can safely allow any process to perform an operation on a shared object without interference from another process
25-Jun		T6.W4.L1	6.2.6	Arbitration	T6.W4.L2	6.2.7	Arbiter With an arbiter, it is possible to choose one of two simultaneous events, safely.
02-Jul		T6.W5.L1	6.2.8	Synchronisation With a synchronizer, one process can be forced to stop and wait for a signal from another.	T6.W5.L2	6.2.9	Deadlocks

09-Jul		**T6.W6.L1**	6.5.1	Most individual human interactions are modelled by an action loop in which a performer delivers a condition satisfying a customer. The loop has a requestor (A) and performer (B) and four time segments culminating in request, promise, delivery and acceptance. Before the loop starts, the condition is not true; when it completes, the condition is true. CSCW - Computer Supported Cooperative Work HCI - Human Computer Interaction CC - Concurrency Control	**T6.W6.L2**	6.5.1	Carried over.
16-Jul		**T6.W7.L1**		**Examination**	**T6.W7.L2**	n/a	Evaluation

Programming Languages for Key Stage 2, 3, 4 and 5

Key Stage 2

Scratch

Scratch is an eminently suitable language to engage 9, 10 and 11 year olds. It is being used successfully at Key Stage 2 in many primary schools across England already. The evidence suggests that primary school teachers responsible for delivering IT lessons acquire knowledge and skill programming in Scratch quite quickly. The evidence also suggests that students find Scratch fun and engaging. The Scratch website - scratch.mit.edu - reports that their website contains over 500, 000 projects encompassing over 12, 000, 000 scripts uploaded by students already. The website supports an online community for educators, ScratchEd, who wish to learn Scratch. Scratch supports creating stories, games and animations thereby catering for a range of student interests and attitudes. The Scratch website allows students to upload their creations so that they may be shared on the Web.

Scratch projects can sense – and respond to – things going on in the world outside the computer, on which Scratch is running, by connecting a PicoBoard - see Figure 1. The PicoBoard is a replacement for the Scratch board which is no longer available,

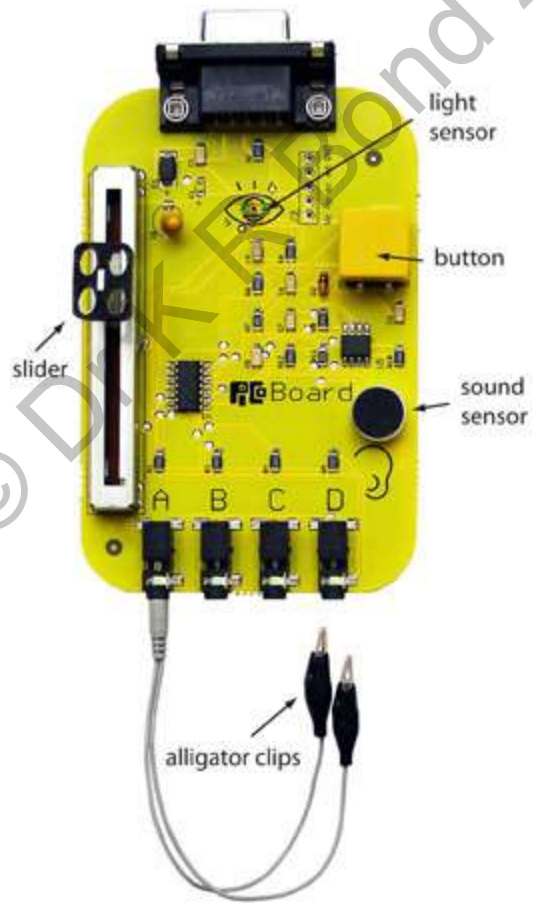


Figure 1 : PicoBoard

It has the following sensors:

1. Sound
2. Light
3. Slider position
4. On/off button
5. 4 resistance sensors

The PicoBoard is available from the Playful Invention Company:

http://www.picocricket.com/index.html

Playful Invention Company
2075 University St., Suite 1208
Montreal, QC H3A 2L1
Canada

Phone: (514) 282-4994
Fax: (514) 313-5521

Unfortunately, they do not have a European distributor. Therefore, the board has to be shipped from the States at a shipping cost of approximately $70 USD. Adoption of Scratch by both primary and secondary schools should develop a market for the PicoBoard in the UK making European distribution a possibility.

Scratch and the Arduino Board

The Arduino - arduino.cc - is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). A programming interface for Scratch should be available shortly.

Figure 2 ***Arduino Duemilanove 328***

To use arduino in a primary environment would require that the arduino be packaged so that the interface presented to students was robust.

The arduino duemilanove is available from Cool Components - www.coolcomponents.co.uk - priced at £55. Figure 3 shows the starter kit available from Cool Components.

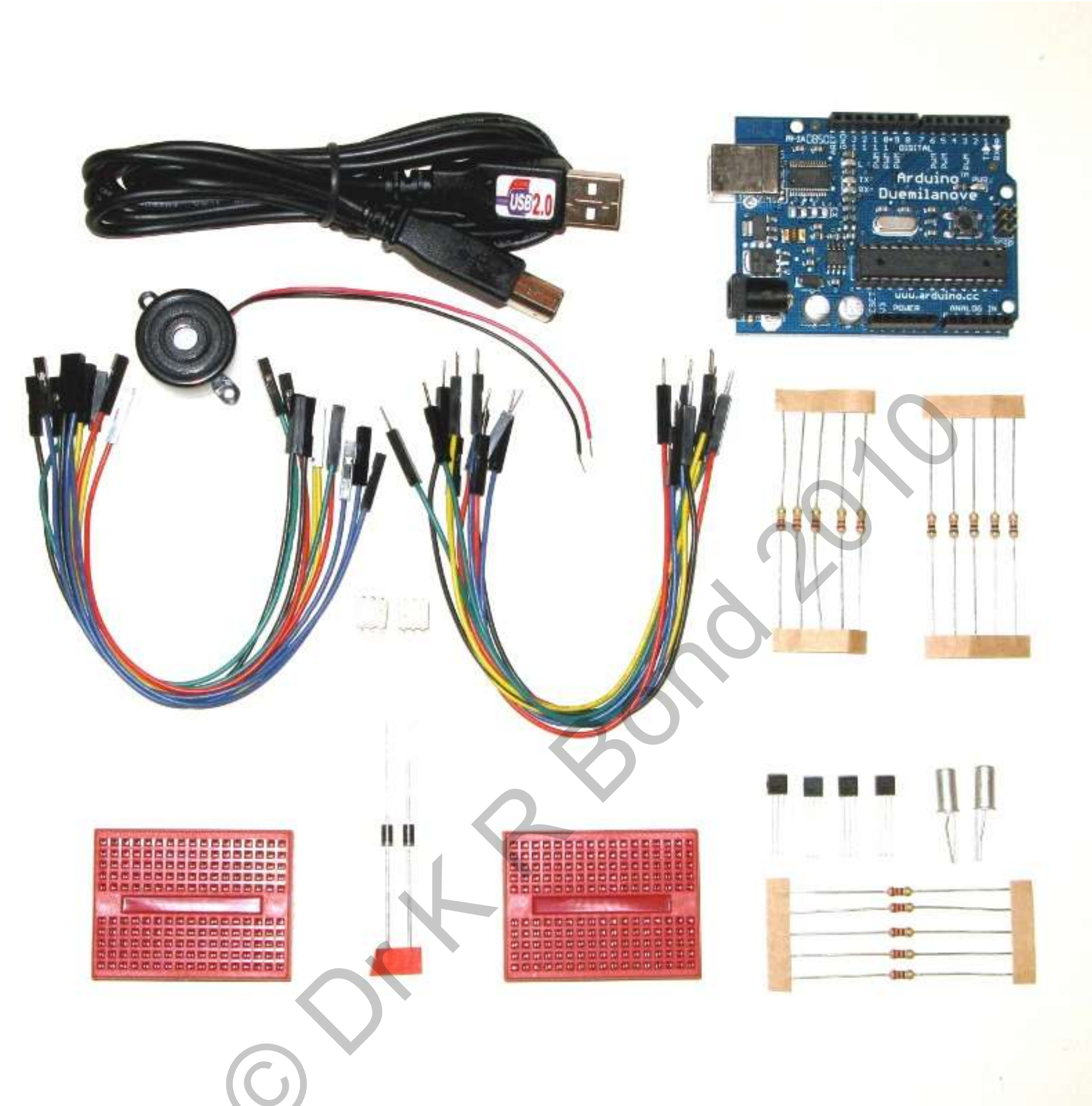


Figure 3 Arduino duemilanove Starter kit

Yenka

Program flowcharting is an alternative approach which students find easy to use. Yenka Programming is available from Crocodile Clips Ltd.-

Crocodile Clips Ltd

43 Queensferry Street Lane

Edinburgh EH2 4PF

Scotland, UK

It is a program flowcharting tool that allows students to control either human characters or on-screen animations. It allows the teaching control, starting with the basic concept of a sequence of steps, and moving on to loops, variables and functions. School site licence varies from £300 to £600 depending on the size of the school. Home use is free. Tutorial support is available from the company's web site.

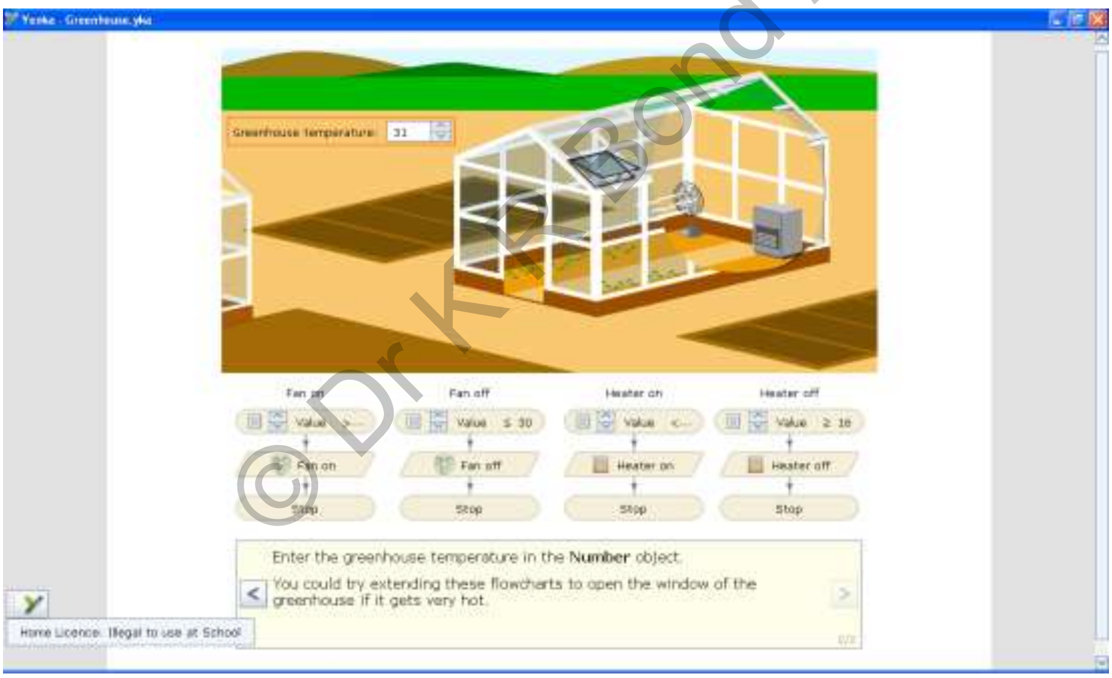


Figure 4 Virtual greenhouse

Figure 4 shows a project using Yenka programming to control a virtual greenhouse system.

Key Stage 3

Research has shown that storytelling is a powerful paradigm. This has sparked interest in creating programming environments that support storytelling. Research also shows that young people engage readily in game-based activities. Learning based on problem solving situated in a games-based environment has a sound pedagogy. Environments in which storytelling and games-based problem solving, are made possible through animations and multimedia are particularly well received by young students.

Scratch

Scratch is also eminently suitable language to engage 11, 12 and 13 year olds. It is being used successfully at Key Stage 3 in many secondary schools across England already. The evidence suggests that students find Scratch fun and engaging. However, in time it may be more appropriate to use an enhanced version of Scratch called BYOB (Build Your Own Blocks) that is currently under development. By constructing their own blocks in Scratch users can learn about important and powerful programming concepts, such as

- defining procedures and functions
- passing parameters
- procedure/function specific variables
- recursion
- "atomicity".

Figure 5 shows an example of recursion in Scratch BYOB.

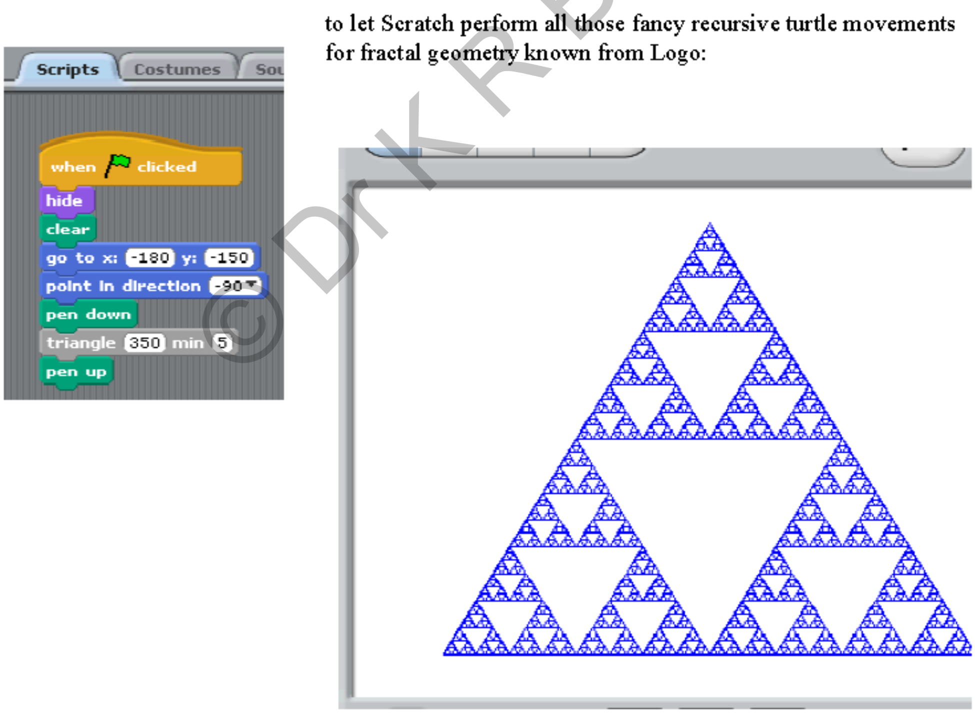


Figure 5 An example of recursion created in Scratch BYOB

Game Maker 7

Game Maker allows computer games to be created without the need to write a single line of code. It uses easy to learn drag-and-drop actions to create professional looking games within very little time. Games can be made with backgrounds, animated graphics, music and sound effects. 3D games are supported. The built-in programming language may be used once the basics of game making have been understood Game Maker can be used free of charge.

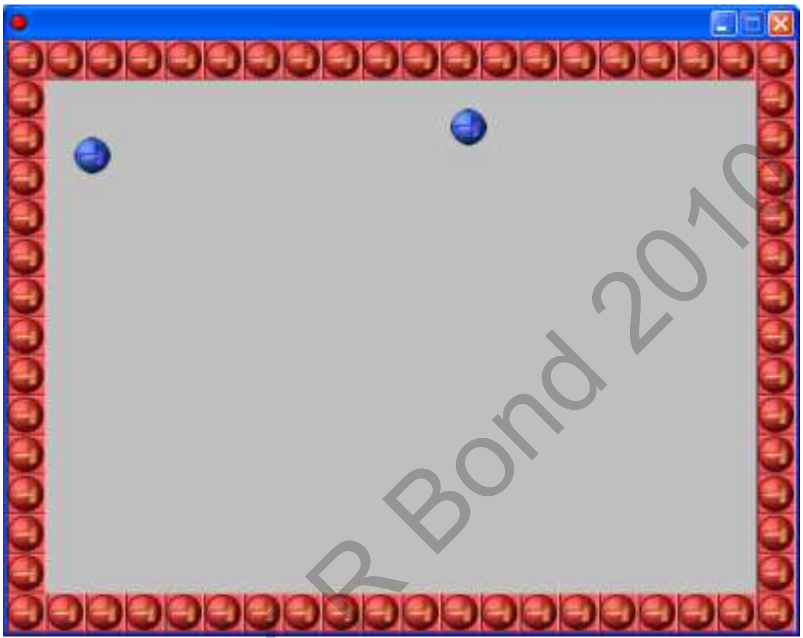


Figure 6 Example of game produced with Game Maker

Scheme

Scheme is based upon Lisp and Lambda calculus but offers a very gentle introduction into how to design of programs. The PLT Scheme system - http://www.plt-scheme.org/ - is free and is accompanied by a free online book *How to Design Programs* . Graphical output is supported.

Bootstrap is a curriculum for 1-12 year old students based around DrScheme which is part of PLT Scheme. It teaches programming through the media of images and animations. It consists of nine 90-minute lessons for delivery once a week.It has a course booklet - http://www.bootstrapworld.org.

Figure 7 shows an example of the materials making up this course.

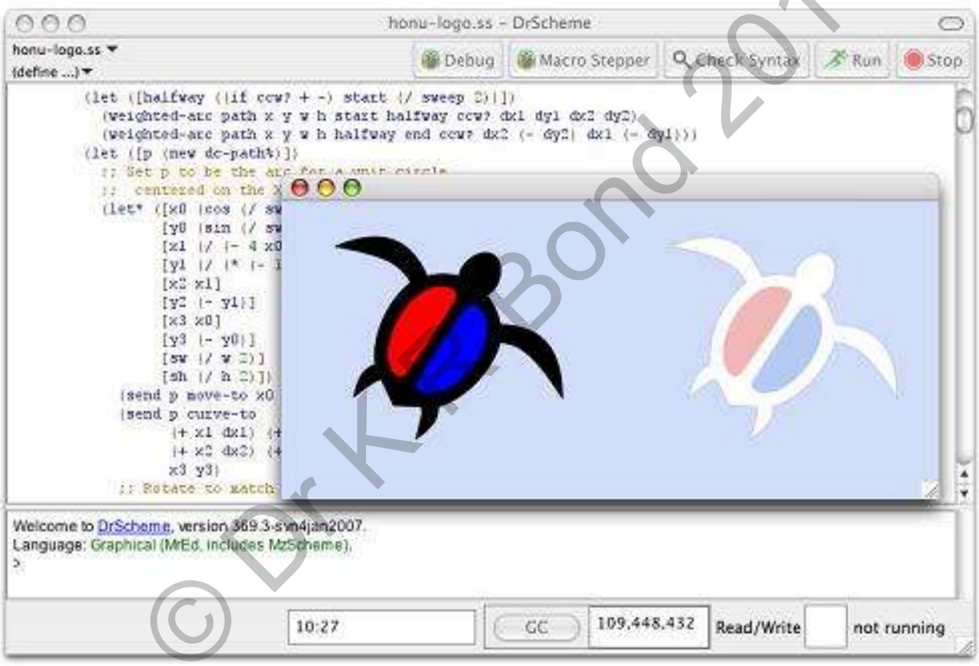


Figure 7 Bootstrap course example

Figure 8 shows the frontcover of another online resource for teaching programming based on DrScheme.

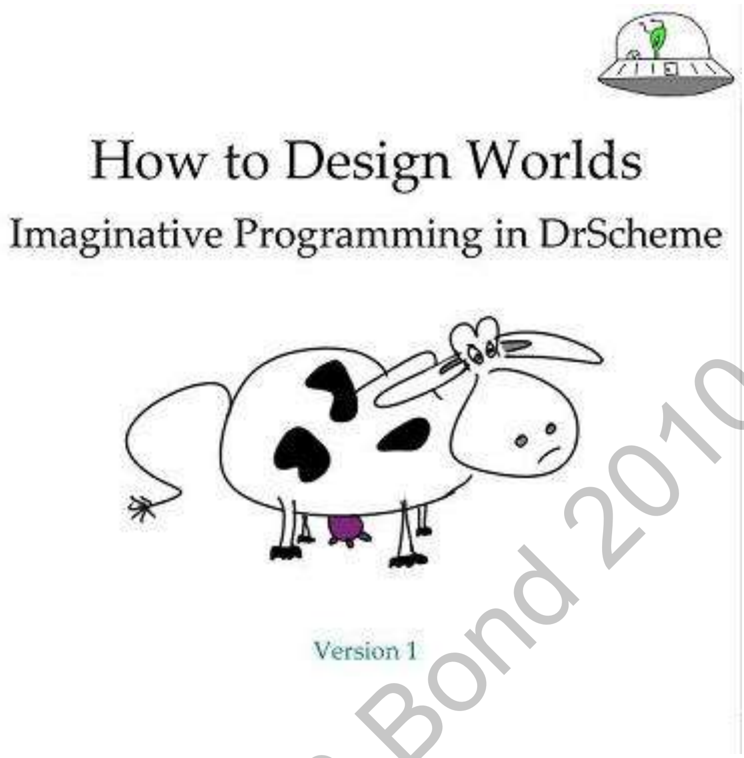
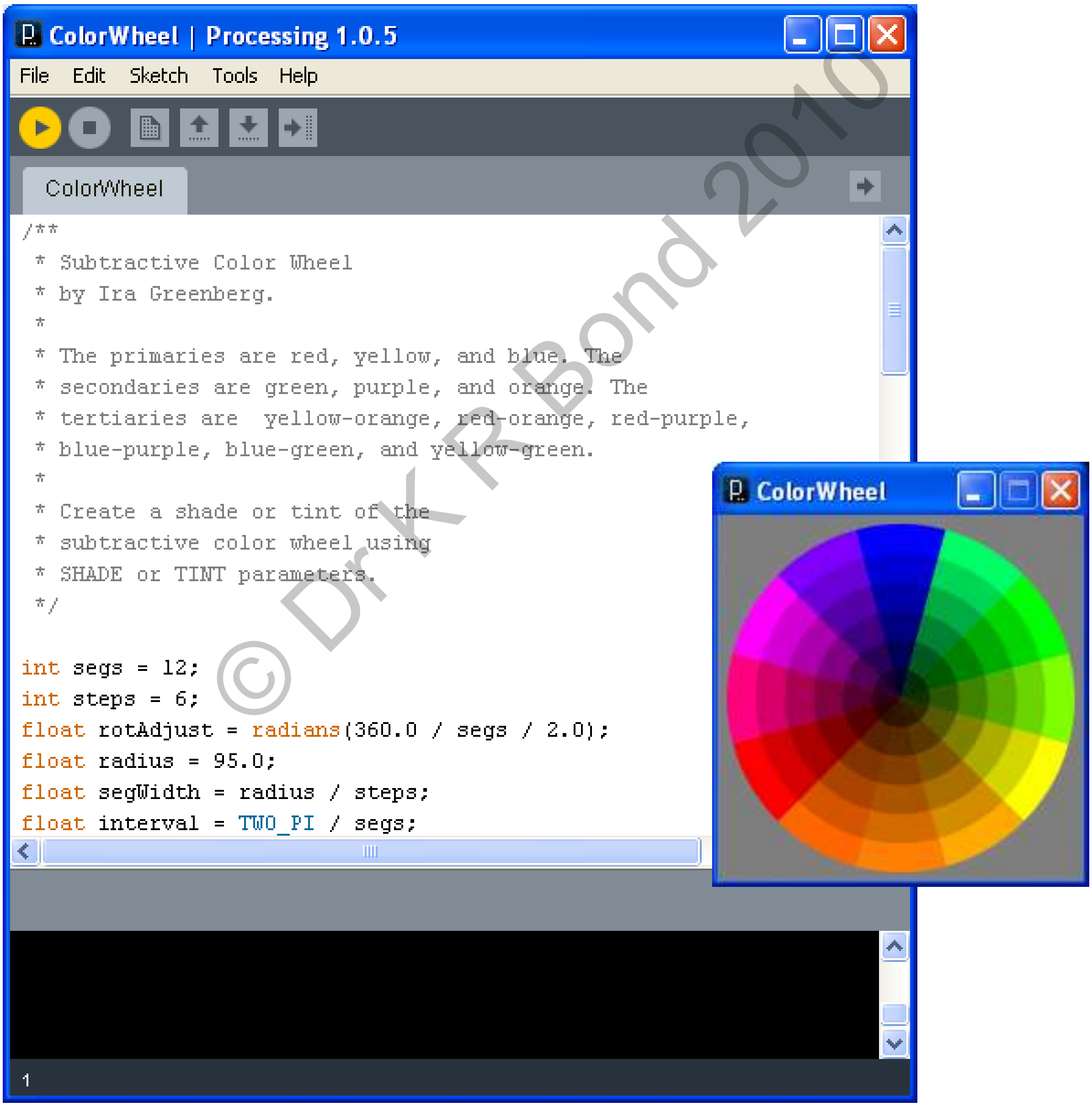


Figure 8 DrScheme course

Processing

Processing - **http://processing.org/** - is an open source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. Processing is a simple programming environment that was created to make it easier to develop visually oriented applications with an emphasis on animation and providing users with instant feedback through interaction. It is created to teach fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool. It comes with numerous example programs which illustrate many of the concepts principles covered in the BOK - Figure 9.



Processing consists of

- The Processing Development Environment (PDE). This is the software that runs when you double-click the Processing icon. The PDE is an Integrated Development Environment (IDE) with a minimalist set of features designed as a simple introduction to programming or for testing one-off ideas.
- A collection of functions (also referred to as commands or methods) that make up the "core" programming interface, or API, as well as several libraries that support more advanced features such as drawing with OpenGL, reading XML files, and saving complex imagery in PDF format.
- A language syntax, identical to Java but with a few modifications.

It is very easy to use and Java applets are generated using the easy to use export command. Figure 10 illustrates how easy Processing is to use. The example program consisting of one statement immediately produces the sloping line when the program is run.

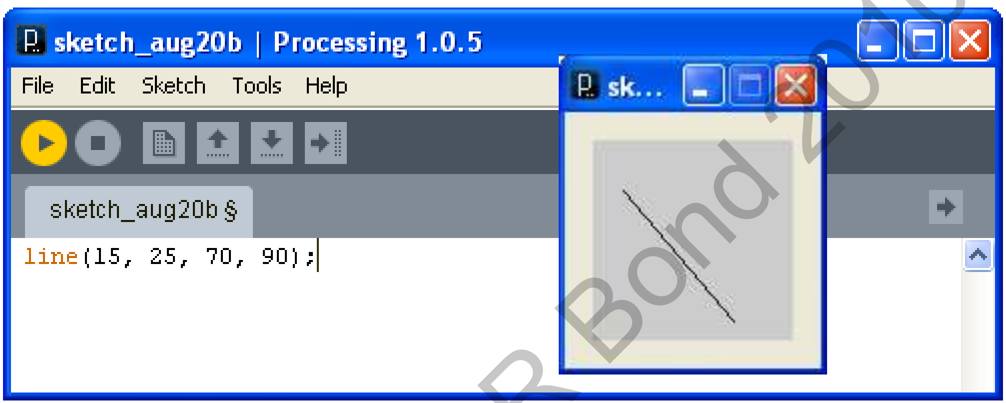


Figure 10 First program in the Processing language

Processing and the Arduino Boards

The *Processing* programming language may be used to control Arduino boards using the special purpose *Arduino* programming language based on processing.

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The *Arduino* programming language is an implementation of *Wiring*, a similar physical computing platform, which is based on the Processing multimedia programming environment.

Mobile Processing Language

The Processing language is at the heart of *Mobile Processing* which is a programming environment for writing mobile phone software. Use of the Mobile processing language would be particularly apposite given the interests of the target audience.

Alice

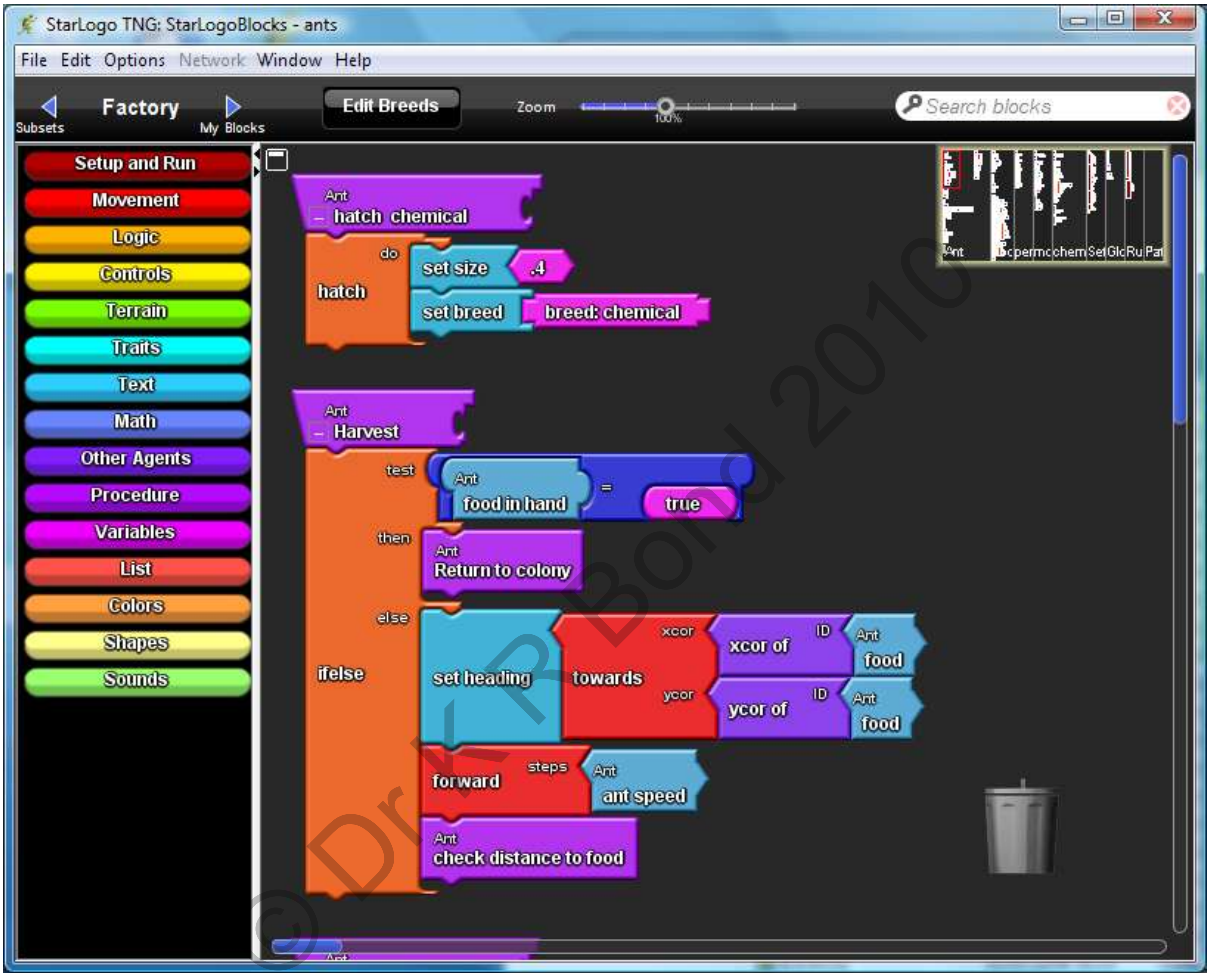
Alice is an innovative 3D programming environment that makes it easy to create an animation for telling a story, playing an interactive game, or a video to share on the web. Alice is a teaching tool for introductory computing. It uses 3D graphics and a drag-and-drop interface to facilitate a more engaging, less frustrating first programming experience. It is free and can be downloaded from Alice.org..

Alice is a teaching tool designed as a revolutionary approach to teaching and learning introductory programming concepts. The Alice team has developed instructional materials to support students and teachers in using this new approach. Resources include textbooks, lessons, sample syllabuses, test banks, and more.

A criticism of Alice 2.0 was that it did not support student transitioning to Java programming. However, this issue has been addressed in Alice 3.0 which has just been released as a beta version. Version 3.0 will let students create animated movies and games with new characters from The Sims video games and will teach advanced users the Java programming language in the process.

StarLogo TNG

TNG stands for "The Next Generation". StarLogo TNG supports parallel programming and simulation. Is like Scratch in that it has a graphical interface where language elements are represented by coloured blocks that fit together like puzzle pieces. It can be used for building games. It uses 3D graphics (unlike Scratch) enabling more compelling and rich games and simulation models to be made. It can be used as a tool to create and understand simulations of complex systems.



The StarLogo TNG game curriculum unit uses computer game design as the motivation and theme to introduce programming to middle or high school students. StarLogo TNG is The Next Generation ofStarLogo modeling and simulation software. Students and teachers use SL-TNG's agents-based programming and 3-D graphics to create and understand simulations and complex systems. Each 1.5 hour lesson includes a mini-lesson to introduce new programming commands and one or more programming exercises to practice using those commands to design a game play element. Ideally, students continue working on programming activities on their own for 30 min to 1 hour outside of class time. Over the course of 10 lessons, students gain the programming knowledge to develop their own "Treasure Hunt" game, a complex system that includes multiple agents and first person game play. They also learn programming basics such as the concept of a forever loop, Boolean logic used in if / then statements, procedures and abstraction, and using variables.

Ant Colony Simulation

Greenfoot

Greenfoot is a software tool designed to let beginners get experience with object-oriented programming. It supports development of graphical applications in the Java™ Programming Language. Figure 12 shows the Greenfoot IDE and Figure 13 its editor. Greenfoot is free and available from www.greenfoot.org.

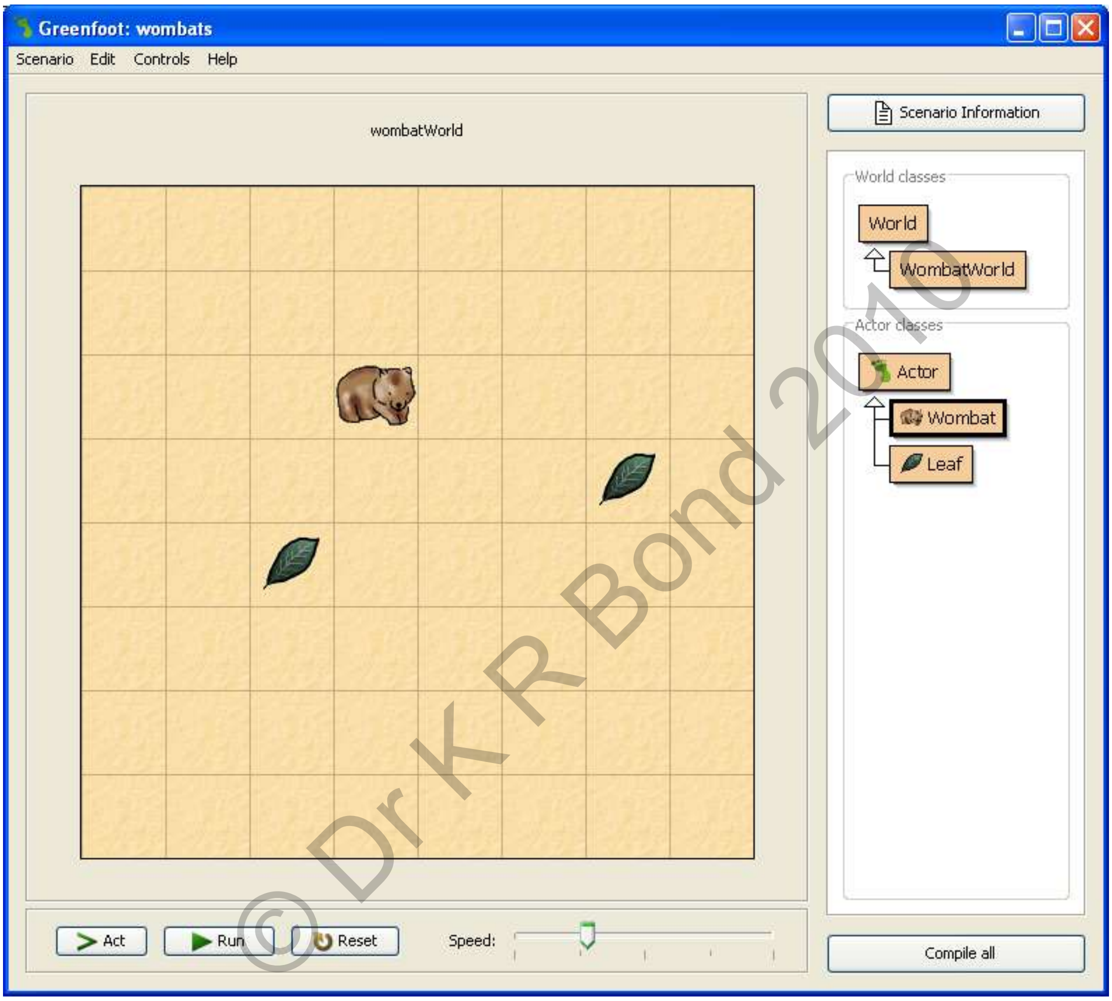


Figure 12 Greenfoot IDE

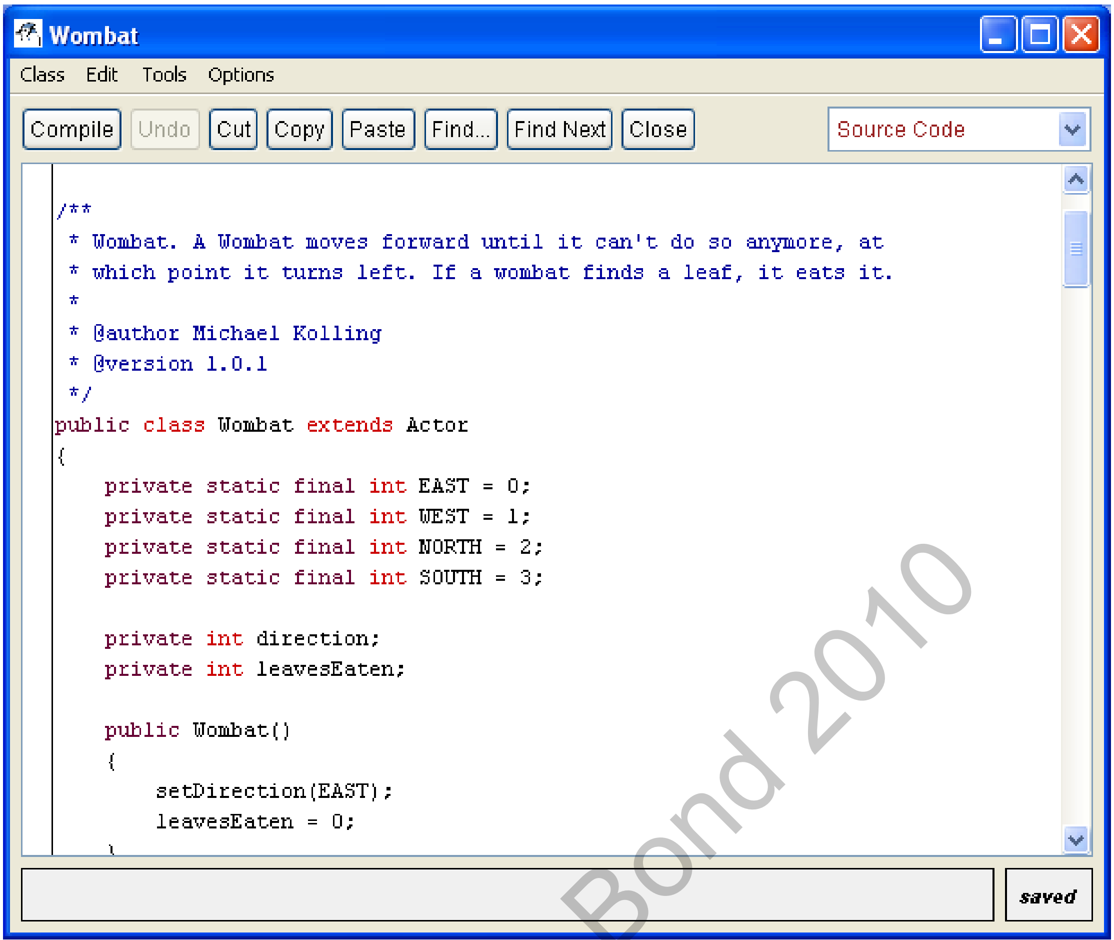


Figure 13 Greenfoot editor

Blackfoot

Blackfoot is a software tool designed to let beginners get experience with object-oriented programming. It supports development of graphical applications Object Pascal Programming Language. At present it requires Embarcadero's Delphi DCC32 compiler but a version based on the FreePascal project is planned for the near future. Figure 14 shows the Blackfoot IDE and Figure 15 its editor. Blackfoot is free and available from Greg Clark from September 20th 2010.

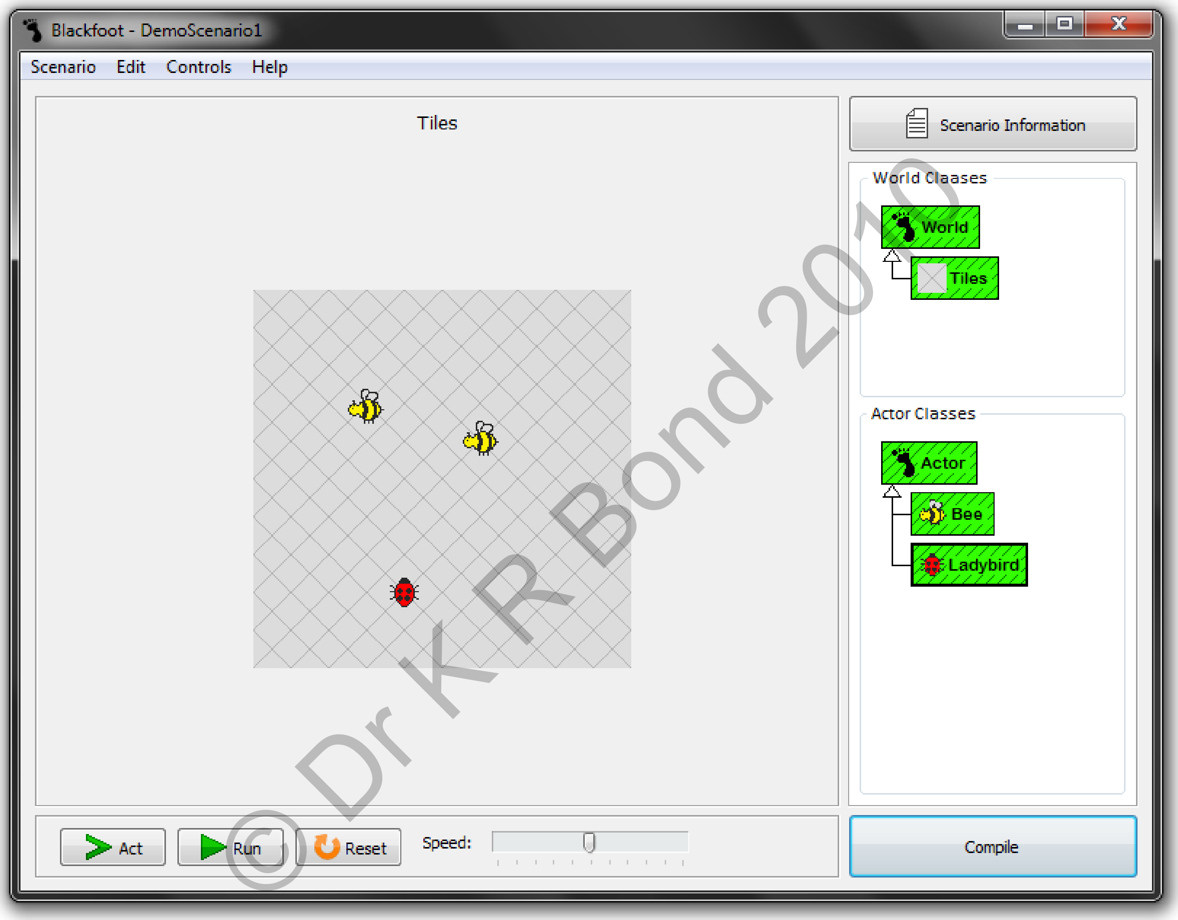


Figure 14 Blackfoot IDE

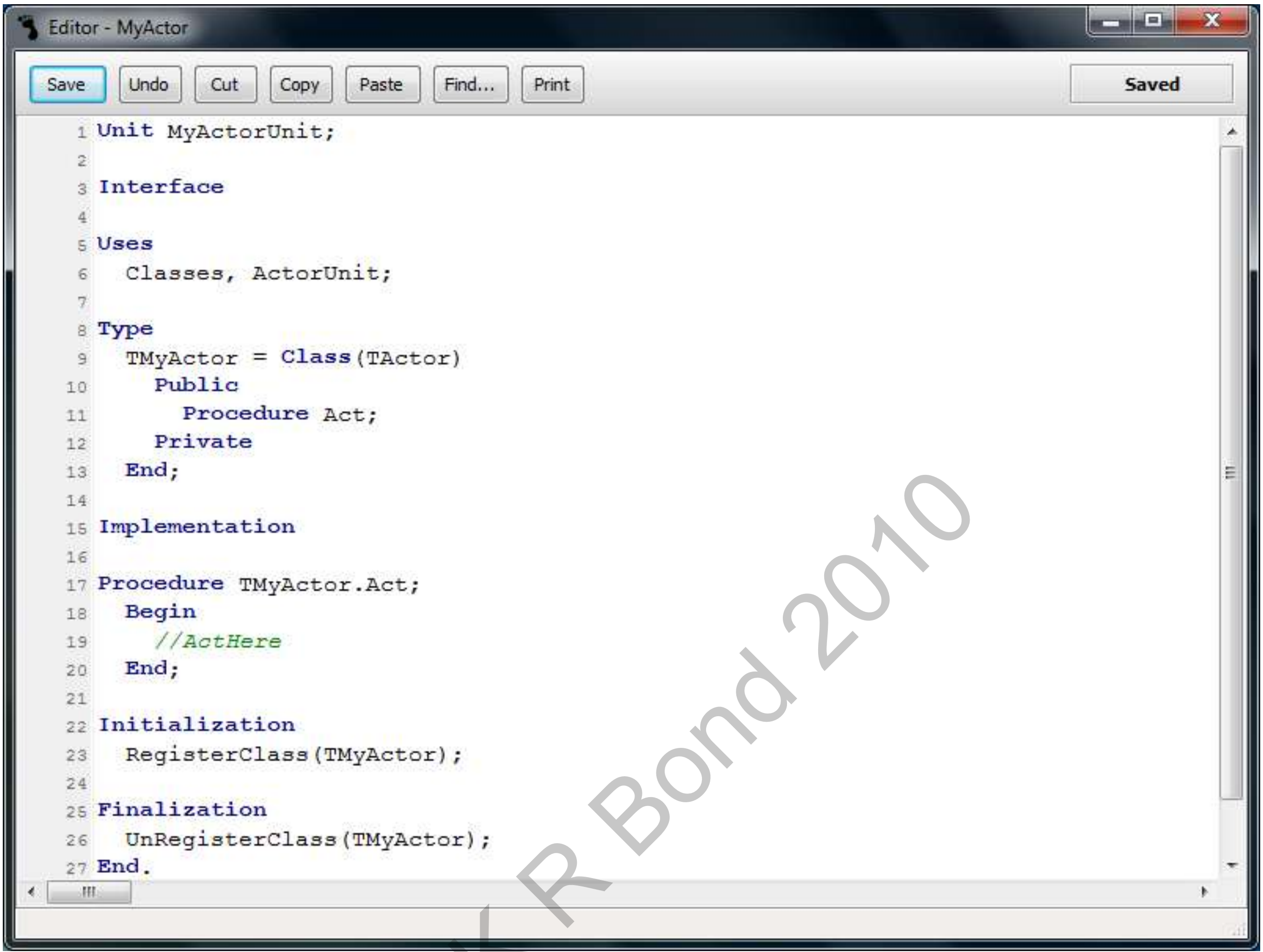


Figure 15 Blackfoot Editor

Visual Basic and VBA

Students find VB and VBA easy to use. VB and VBA can be used in conjunction with Microsoft's Office Suite which is useful.

Lego Mindstorms

Lego Mindstorms uses a graphical programming language, Lego Mindstorms Edu NXT. However, it is expensive and requires careful management in a classroom environment as the kit is easily damaged or disassembled - 16 robots would cost approximately £1600. Using Lego Mindstorms robots has implications for maintenance support which could be a significant overhead for a school.

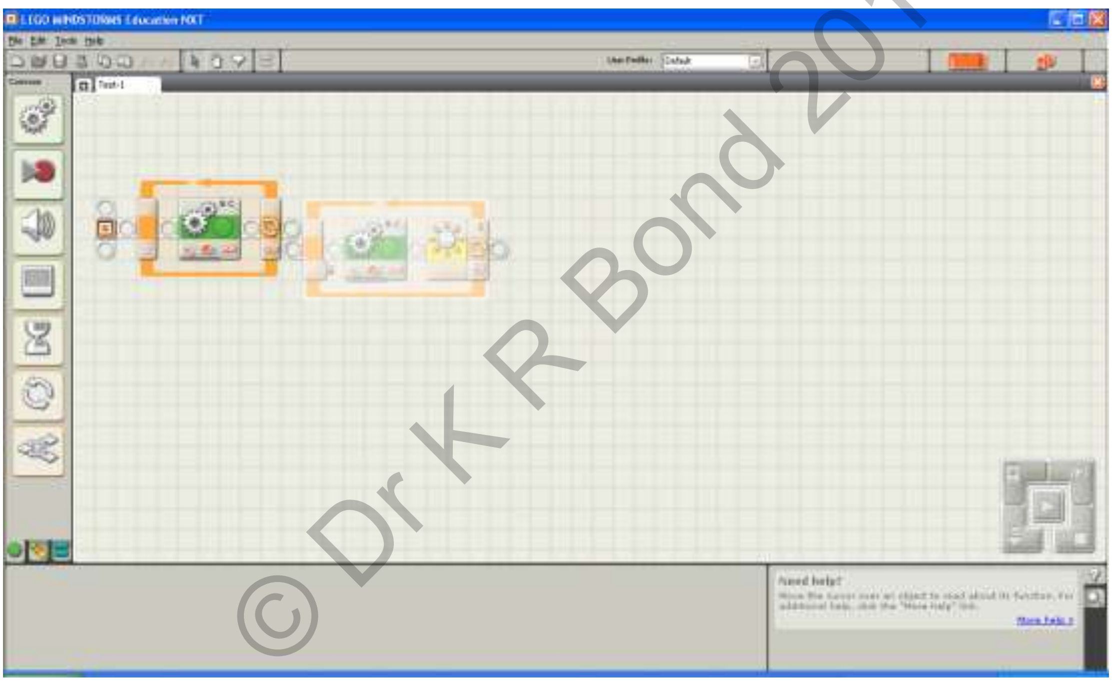


Figure 16 Lego Mindstorms programming environment

Flash

Flash is a tool for creating interactive and animated Web sites!

Flash is a multimedia graphics program specially for use on the Web
Flash enables you to create interactive "movies" on the Web
Flash uses vector graphics, which means that the graphics can be scaled to any size without losing clarity/quality
Flash does not require programming skills and is easy to learn

JavaScript

Object-oriented scripting language that is relatively easy to use and freely available. Supported by University of Washington Benefit, Fluency with IT course, (http://courses.washington.edu/benefit/FIT100/).

Assembly Language Programming

University of Hertfordshire, Computer Science Department Machine Simulator

Simple simulator which illustrates machine architecture, the fetch-execute cycle and assembly language programming - http://homepages.feis.herts.ac.uk/~msc_ice/fe2/. This can be run within a browser connected to this site or downloaded and run within a browser on a local machine.

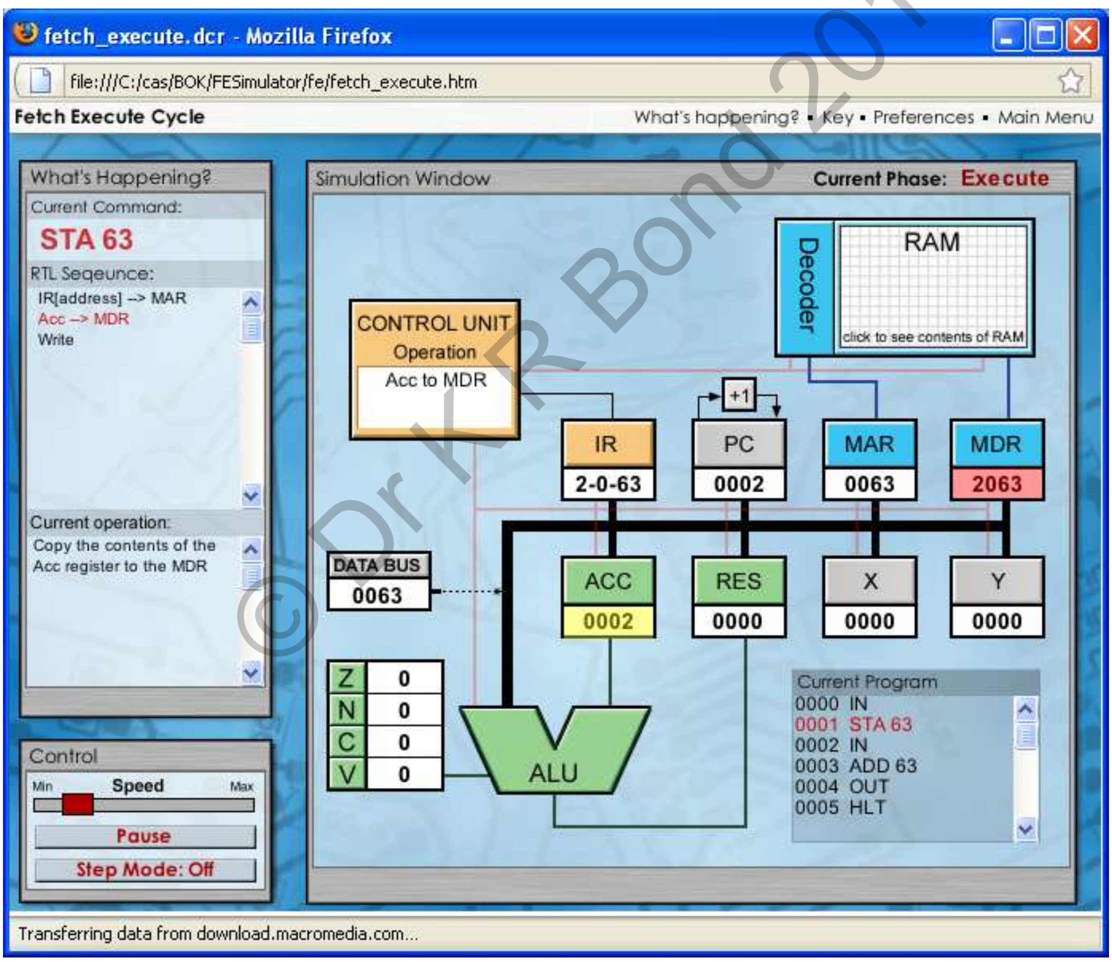


Figure 17 Fetch-execute cycle (need permission from CS deparment Univ. of Hertfordshire to use)

ASMTutor

ASMTutor is an assembly language simulator that can be used to teach assembly language and machine architecture. ASMTutor is supplied with example programs. It is available from Educational Computing Services Ltd - www.educational-computing.co.uk.

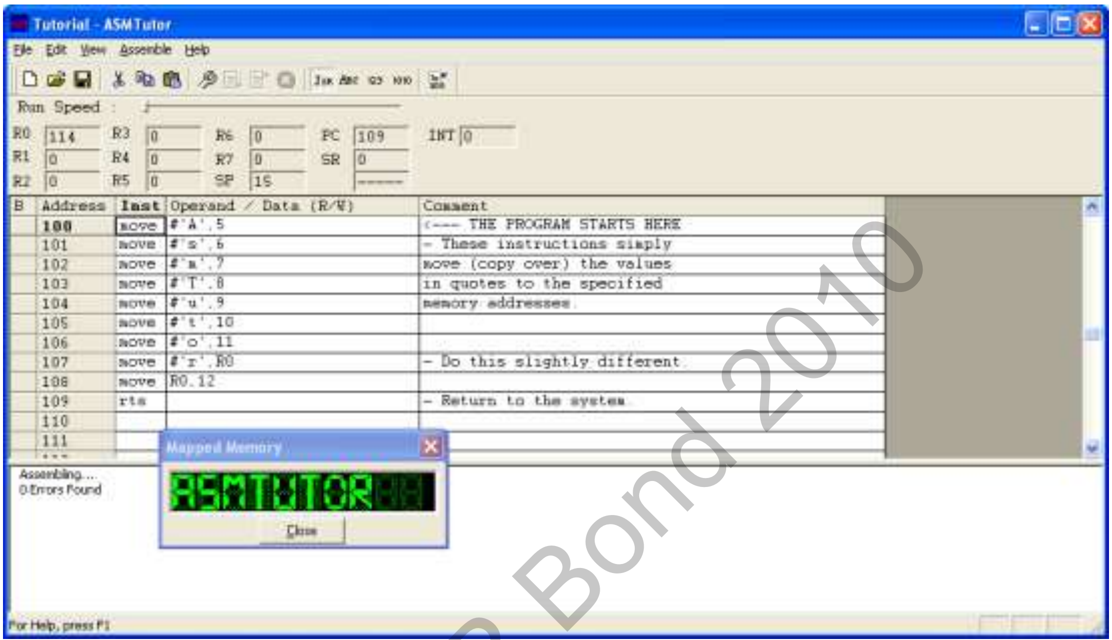


Figure 18 ASMTutor user interface

Key Stage 4

Processing

The Processing programming language is a very suitable language for this Key Stage. Its support for programming the cheap Arduino microcontroller boards as well as support for creating mobile phone applications makes it an attractive option. Processing is easy to learn as well as being very powerful and well supported by affordable and excellent textbooks - e.g. Learning Processing: A beginner’s guide to programming Images, Animation and Interaction by Daniel Shiffman. It doesn’t suffer from the convoluted syntax of Java, yet supports transitioning to the Java programming language.

Greenfoot

Greenfoot has a sound pedagogy but its requirement for students to program in the Java programming language reduces its attraction. Java’s syntax is too convoluted for the average student. It only supports the object-oriented paradigm and does not allow easy exploration of algorithms as does Processing, Scratch BYOB, Scheme and Delphi. Switching to native Java doesn’t seem appropriate given the steep learning curve of the native language. Java is too hard a language to learn for the average fifteen year-old.

Delphi

Delphi is a high-level, compiled, strongly typed language that supports structured and object-oriented design. Delphi language is based on Object Pascal. It supports event-driven forms-based(windows-based) programming as well as console mode programming. Both programming methodologies can be combined together in one program so that outputs can appear in a console window as well as a forms-based window. Its excellent component library makes the task of generating solutions for a range of applications from database system through networking to OpenGL graphics very easy and certainly much easier than Java. Its syntax is also easier to learn than Java. A feature not found in other modern languages is the notion of sets. Delphi's set type is a collection of values of the same ordinal type. Delphi also benefits from being based on a language, Pascal, with a very long pedigree that was designed for teaching structured programming but which was extended as Object Pascal to support object-oriented programming. Both standard Pascal and Object Pascal are governed by ISO standards. This means that there is excellent support for Delphi, Pascal and Object Pascal with several free systems being available as well as excellent backwards compatibility. Delphi can be used to control the Phidget microcontroller for sense and control work.

FreePascal is a very versatile version of Pascal with support for Compiling and running pure Pascal applications on the iPhone simulator as well as a cross-compiler for ARM processors used in smart phones.

Phidgets

Phidgets are a set of "plug and play" building blocks for low cost USB sensing and control from a PC. All the USB complexity is taken care of by a robust API. Applications can be developed quickly by programmers using any of the following languages: C/C++, C#, Cocoa, Delphi, Flash AS3, Flex AS3, Java, LabVIEW, MATLAB, Max/MSP, MRS, Python, REALBasic, Visual Basic.NET, Visual Basic 6.0, Visual Basic for Applications, Visual Basic Script, and Visual C/C++/Borland.NET.

Figure 19 Phidget board

Available from Active Robots - http://www.active-robots.com/products/phidgets/

Java

VB

Python

Javascript – can be used to create apps for Apple iphone

PHP

Key Stage 5

Delphi

Java

VB

Python

PHP

Javascript

Computing could be said to draw upon topics from the following subject areas

- Discrete mathematics: logic, graphs, automata theory, information theory, probability, category theory.
- Engineering: computer hardware, transistors, logic gates, robotics, signal processing, control theory.
- Computer peripherals: screen, keyboard, mouse, printer, etc.
- Telecommunications: bandwidth, optical fibres, wireless, networks.
- Philosophy: artificial intelligence, the mind/body problem, philosophical logic.
- Sociology: human/technology interaction and acceptance, team-working.
- Psychology: reasoning, learning, memory, perception, communication, requirements capture.
- Linguistics: speech and language recognition and generation, parsing, semantics.
- Biology: neuroinformatics, brain, biologically inspired computing.

Example Worksheets, Activities, Teaching Guides and Reference Material

This topic is about “1.1.4 Meaning is discerned and acted upon by observers reading pattern” in Data and Information

Consider the truth table for an AND logic gate

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

Now consider how this might be realised in practice.

If logic level 0 is encoded as a voltage of zero volts and logic level 1 as a voltage of five volts we construct a table of voltages as follows:

Input 1	Input 2	Output
Zero	Zero	Zero
Zero	Five	Zero
Five	Zero	Zero
Five	Five	Five

AND gate

Now if the voltages are reinterpreted as follows:

A voltage of zero volts encodes logic level 1 and a voltage of five volts encodes logic level 0, and the voltage table mapped to a logic level table we have an OR gate.

Input 1	Input 2	Output
1	1	1
1	0	1
0	1	1
0	0	0

OR gate

Nothing physical has changed. What has changed is the interpretation that is placed on the observed voltages.

Hence **meaning is discerned and acted upon by observers reading pattern.**

Of course, the AND gate does execute a logical operation because a conscious individual has constructed it to do exactly this and nothing else.

Discrete Mathematics

This topic is about modular arithmetic and an application in error detection

Task: The sum method

- Add together the digits of the following decimal number: 581973471987
- Answer: 69
- Now add together the digits of the answer 69
- Answer: 15
- Now add together the digits of the answer 15
- Answer: 6

Task: The modulo arithmetic method

- Calculate 581973471987 Modulo 9
- Answer: 6
- Sum method:
    - If answer is 9 then replace by 0
    - e.g. 18 → 9 → 0

Now try both methods on other decimal numbers

Why do both methods give the same answer?

Modular arithmetic

N Mod 9	N
Result 0	{0, 9, 18, 27, 36, 45, ...}
Result 1	{1, 10, 19, 28, 37, 46, ...}
Result 2	{2, 11, 20, 29, 38, 47, ...}
Result 3	{3, 12, 21, 30, 39, 48, ...}
Result 4	{4, 13, 22, 31, 40, 49, ...}
Result 5	{5, 14, 23, 32, 41, 50, ...}
Result 6	{6, 15, 24, 33, 42, 51, ...}
Result 7	{7, 16, 25, 34, 43, 52, ...}
Result 8	{8, 17, 26, 35, 44, 53, ...}

Check the value of *Result* for sampled values from the sets of integers, N

N Mod 9 Calculates the Remainder

- Result 1 {1, 10, 19, 28, 37, 46, ...}
- 9 does not divide 10 evenly there is 1 left over
  Similarly, 9 does not divide 19 evenly there is 1 left over
- The same is true of all the numbers in the given set
- We call the 1 left over the remainder
- That is what the Mod operation calculates – the remainder

Check Digit

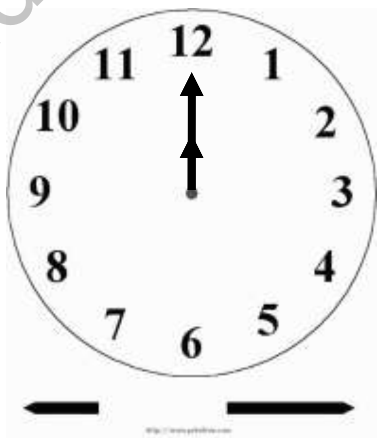
- Let's suppose that we wish to detect if a number has arrived at its destination unaltered
- We know that if we send any of the numbers in the set {3, 12, 21, 30, 39, 48, ...} then **number Mod 9 = 3**
- If we send the number 3 as well as the number and compare then we can detect when the number has been changed into a number not in the given set.

Instruction

Send a long decimal integer number to a neighbour together with its check digit. Get your neighbour to check, as best as he or she can, that the long number has not been corrupted on route. If you write in HB pencil and use a neighbour who is sat several rows of desks away then using a rule that anyone in the path to your neighbour can erase one or more digits and replace with different digits, if they want to, your neighbour should detect most of the time when this occurs.

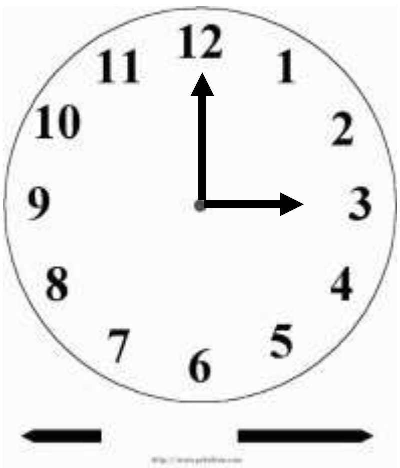
Do you realise that you have been doing modular arithmetic since you could tell the time?

N Mod 12	N
Result 0	{0, 12, 24, 36, 48, 60, ...}
Result 1	{1, 13, 25, 37, 49, 61, ...}
Result 2	{2, 14, 26, 38, 50, 62, ...}
Result 3	{3, 15, 27, 39, 51, 63, ...}
Result 4	{4, 16, 28, 40, 52, 64, ...}
Result 5	{5, 12, 29, 41, 53, 65, ...}
Result 6	{6, 12, 30, 42, 54, 66, ...}
... .	
Result 11	{11, 23, 35, 47, 59, 71, ...}



What time does the clock show after 40 hours?

Where will hands be after 22 hours?



Congruence

N Mod 9 **N**

Result 4 {4, 13, 22, 31, 40, 49, ...}

We say that 13 is congruent to (4 mod 9) and we say that 4 is congruent to (13 mod 9) because the same result is generated when modulo 9 is performed on each.

Error detection when the symbols are just 0 and 1

N Modulo 2	**N**
Result 0	{0, 2, 4, 6, 8, 10, ...}
Result 1	{1, 3, 5, 7, 9, 11, ...}

Single-bit Error Detection

1	1	0	1	0	1	0	0

$\uparrow$

Parity bit

EVEN Parity – even number of 1 bits

7 data bits and 1 parity bit

Summing the data bits modulo 2

(1 + 0 + 1 + 0 + 1 + 0 + 0) Mod 2 = 3 Mod 2 = 1

Result 1 {1, 3, 5, 7, 9, 11, ...}

> Summing all the bits modulo 2 – data and parity – gives the result 0. This indicates that no error has occurred if even parity is being used.

If there is an error in bit 3 (numbering from left)

1	1	1	1	0	1	0	0

$\uparrow$

Parity bit

EVEN Parity – even number of 1 bits

7 data bits and 1 parity bit

Summing the data bits modulo 2

(1 + 1 + 1 + 0 + 1 + 0 + 0) Mod 2 = 4 Mod 2 = 0

Result 0 {0, 2, 4, 6, 8, 10, ...}

Parity bit = 1 1 1 1 1 0 1 0 0

Calculated parity ≠ parity bit therefore an ERROR

> Summing all the bits modulo 2 – data and parity – gives the result 1. This indicates an error has occurred. The output should be 0 for an even parity system with no error

Engineering

This topic is about connecting together logic gates to make useful computing circuits.

Example

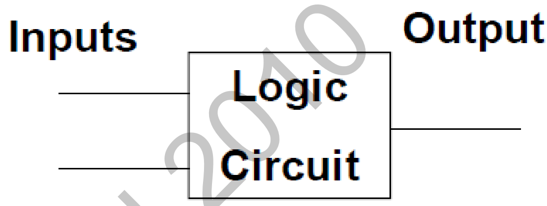
Can we make a logic device which performs input1 + input 2 Mod 2 with input1 and input2 either 0 or 1?

Exclusive OR

Computing

Truth Tables

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

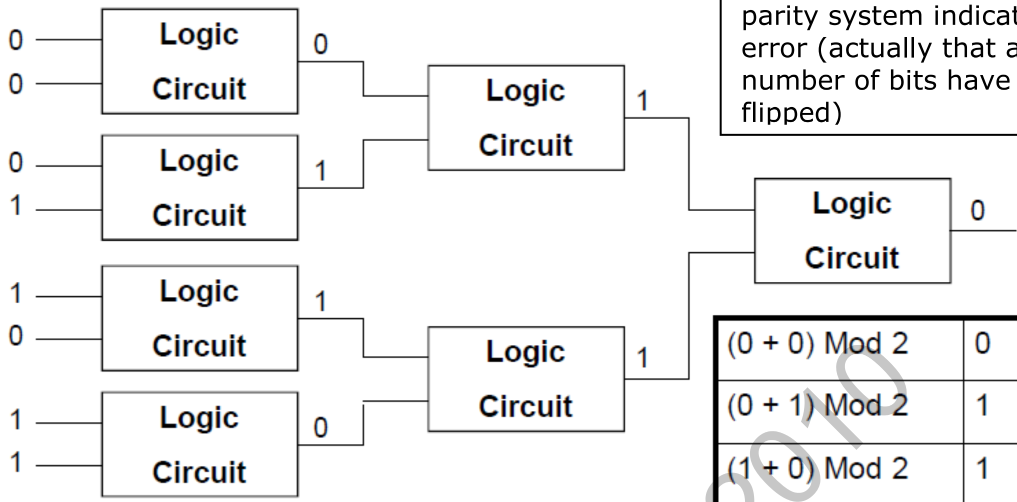


(0 + 0) Mod 2	0	
(0 + 1) Mod 2	1	
(1 + 0) Mod 2	1	
(1 + 1) Mod 2	0	

Parity Checker for Seven Data Bits and One Parity Bit

Combining this logic circuit with others of the same type we have produced a parity bit checker

A final output of 0 in an even parity system indicates no error (actually that an odd number of bits have not been flipped)



(0 + 0) Mod 2	0	
(0 + 1) Mod 2	1	
(1 + 0) Mod 2	1	
(1 + 1) Mod 2	0	

(0 + 0 + 0 + 1 + 1 + 0 + 1 + 1) Mod 2 = 4 Mod 2 = 0

Now try a different correct pattern of eight bits (seven data bits and one parity bit) using even parity. Does this parity checker work?

What difference would using odd parity make to how this collection of logic circuits is used?

Can you design a parity bit generator that works with 7 data bits?

Telecommunications

This topic is about understanding bandwidth.

The 16 by 16 chequered grid shown in Figure 1 represents a section of the screen of a mobile phone. The mobile phone screen has a screen size of 128 by 128 pixels (squares). This is known as the resolution of the screen.

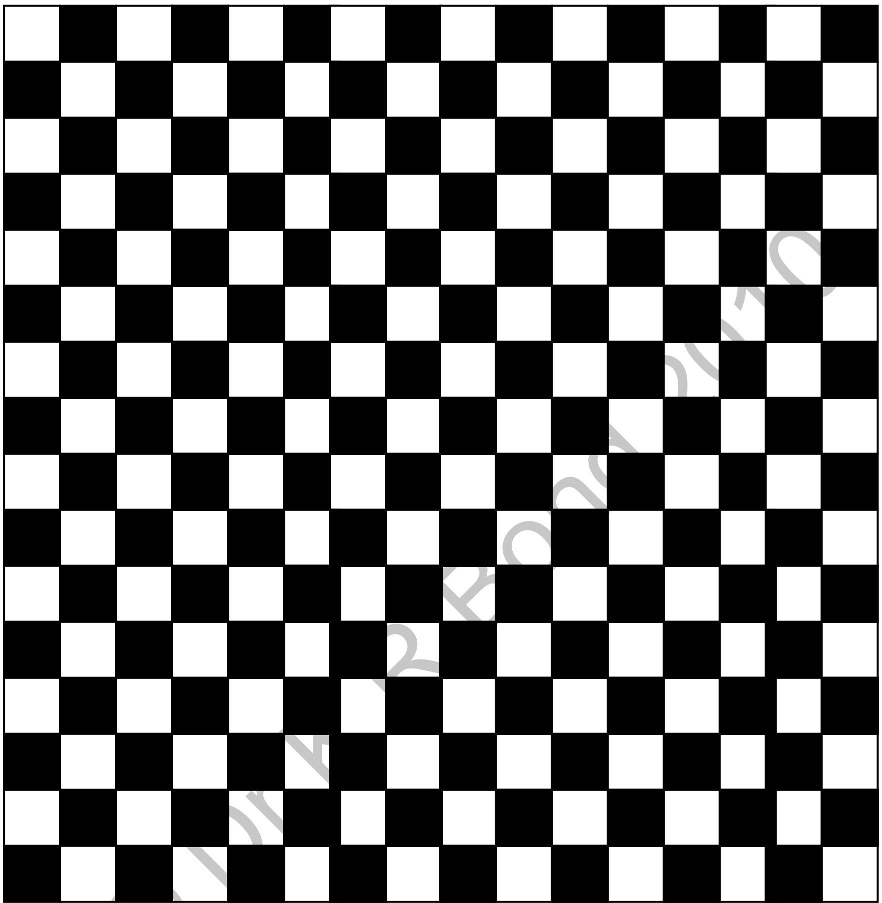


Figure 1

Let's suppose for the moment that the entire mobile phone screen consists entirely of this 16 by 16 grid and that each "square" is either black or white. How could we code the black or white state of the 16 x 16 grid using just the digits 0 and 1?

(See the next page for the answer).

The answer is shown in Figure 2 below. A 1 in a square of the grid of Figure 2 corresponds to a white square in Figure 1 and a 0 in a square of the grid of Figure 2 corresponds to a black square in Figure 1. The grid in Figure 2 consists of 16 x 16 bits (a bit is a single binary digit, a binary digit is restricted to the symbols 0 and 1) or 256 bits in total.

1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Figure 2

Let's suppose that the black and white image of Figure 1 needs to be refreshed 100 times a second. This means that the 256 bits encoding the image must be sent to the screen 100 times a second. In one second 100 x 256 bits are delivered to the mobile phone's screen or, multiplying out, 25600 bits per second.

If you were to be able to see these bits moving to the screen from somewhere inside the mobile phone they would resemble Figure 3 which represent a series of voltages which alternate between zero volts to 5 volts repeatedly. The voltage level 5 volts is used to represent binary digit value 1 and the voltage level 0 volts is used to represent the binary digit value 0.

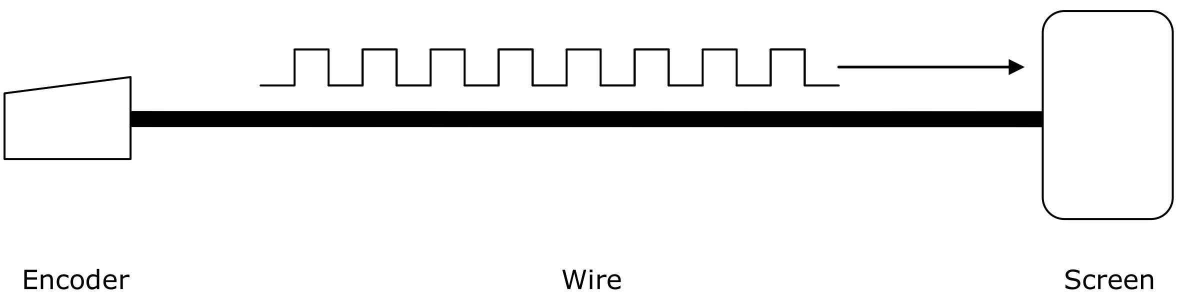


Figure 3

Bandwidth

For the 16 x 16 black and white screen the wire must be capable of carrying 25600 bits per second. Bandwidth can be measured in bits per second or bps. So for the 16 x 16 black and white screen the wire must support a bandwidth of 25600 bits per second.
Now if the mobile phone screen actually comprises a 128 x 128 black and white screen then the bit rate that the wire supports needs to be at least 128 x 128 x 100 bps or 1638400 bps.

If we use the unit **K** to represent 1000 and **M** 1000000 then we can measure bit rate in **K** units or **M** units, i.e. **Kbps** or **Mbps** meaning kilobits per second and megabits per second respectively. Our 25600 bps becomes 25.6 Kbps and our 1638400 bps becomes 1638.4 Kbps. Bandwidth can also be expressed in Hertz (Hz) which is a unit used to measure the number of oscillations per second of an oscillation (e.g. 0 volts then 5 volts then 0 volts then 5 volts and so on). For technical reasons, when bandwidth is expressed in Hz the bandwidth is taken to be twice the bit rate, i.e. 25.6 Kbps bit rate becomes a bandwidth of 51.2 KHz.

There is a direct relationship between **data rate (bit rate)** and b**andwidth**. The **greater the bandwidth** of the transmission system, the **higher is the data rate** that can be transmitted over that system.

If the data rate of the digital signal is **W** bps then a very good representation can be achieved with a bandwidth of **2W** Hz.

The Apple iPhone 3GS has a screen resolution of **360 x 640**.

What bit rate must this screen work at if the screen has only black and white pixels (which it doesn't but we will address this presently) and the screen refreshes 100 times a second?

The Apple iPhone 3GS has a colour depth of **24** bits. This means that 24 bits are used to encode the colour of each pixel.

Recalculate the bit rate for the Apple iPhone 3GS, resolution 360 x 640, colour depth 24 bits.

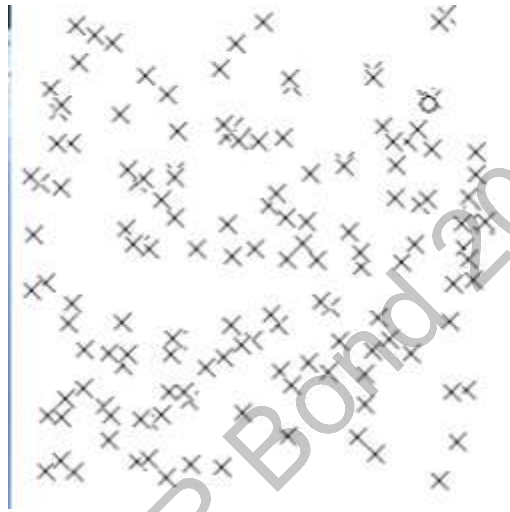
Psychology

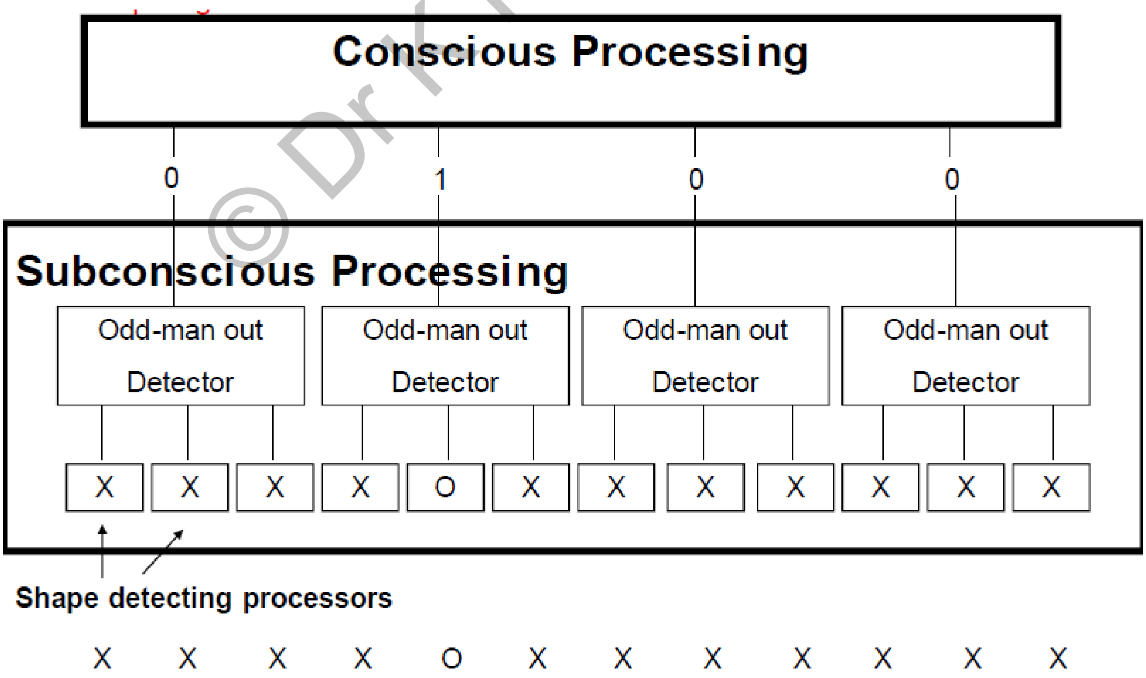
This topic is about visual perception

Basketball Video - http://www.youtube.com/watch?v=2pK0BQ9CUHk
Why was it so difficult?

Program displays shapes X 's and O's – Program 1

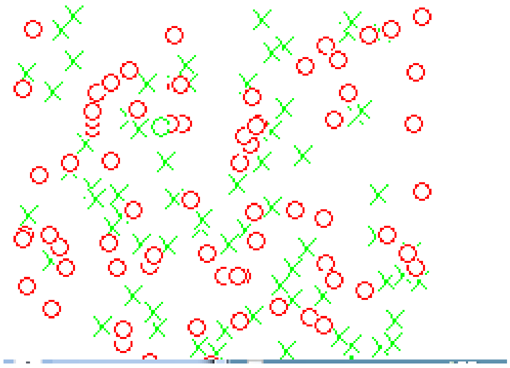
- Count the number of times that an O appears (one O in a sea of crosses)
- It doesn't matter how many Xs there are, people say that the O just pops out.
- Pop-out is a sign of parallel processing

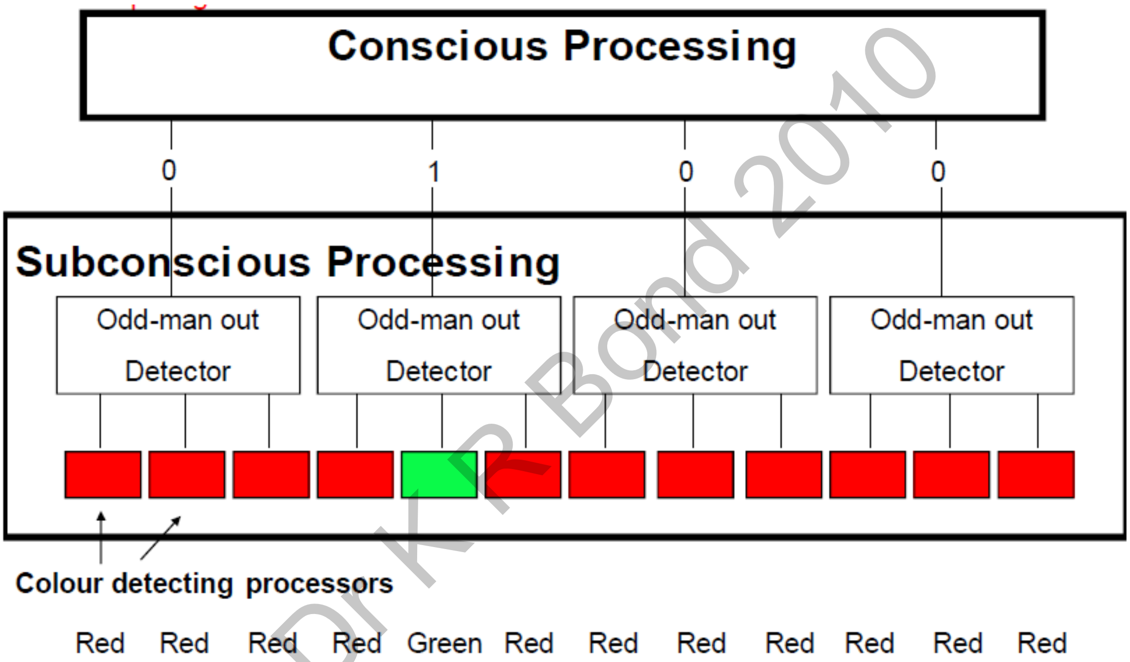




- The raw data and computational steps are sealed off from our awareness

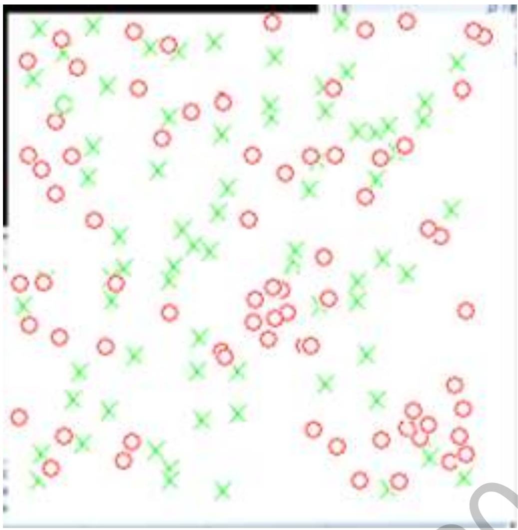
Green O pops out from a sea of red O's – Program 2

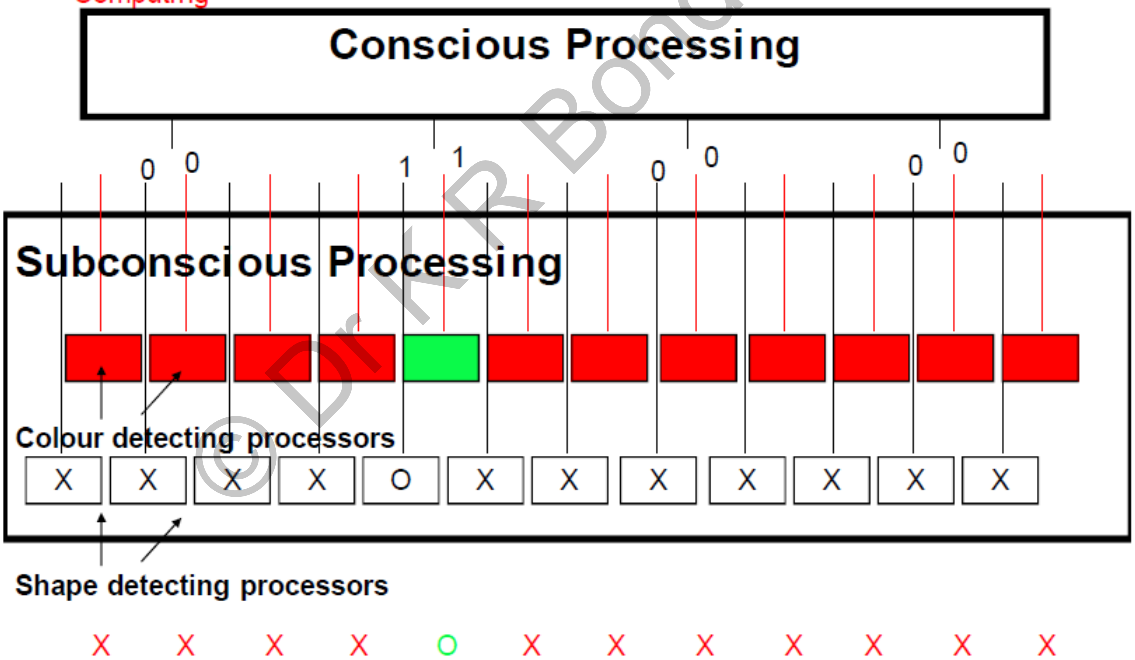




- The raw data and computational steps are sealed off from our awareness

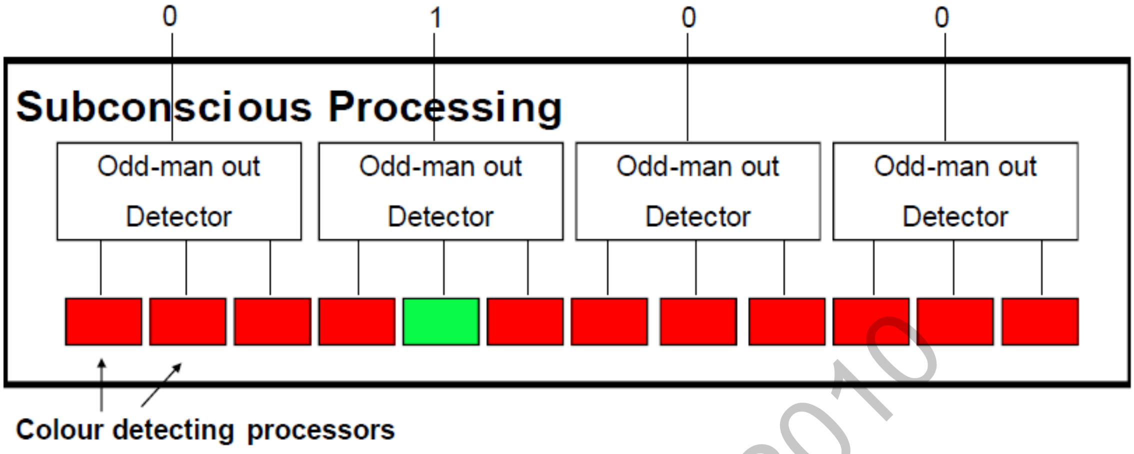
Green O does not pop out from a sea of green Xs and red O’s – Program 3





- Find a letter that is both green and an O
- The letter sits somewhere in a mixed sea of green X’s and red O’s

- **The thousands of processors tiled across the visual field are too stupid to calculate conjunctions of features:**
  - **A patch that is green and curved**



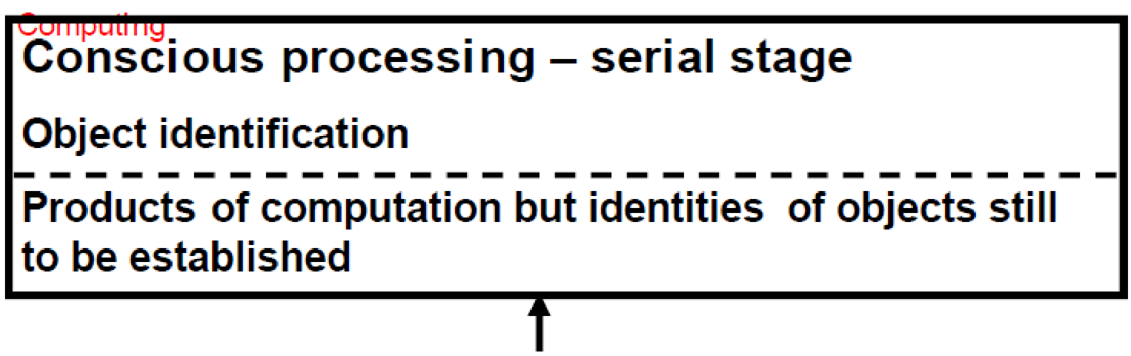
Spotlight of attention

Unconscious parallel processing (in which many inputs are processed at the same time, each by its own mini-processor) can only go so far. Parallel processing does what it can and passes along a representation from which a more cramped and plodding processor must select the information it needs.

It would be impossible to sprinkle conjunction detectors at every location in the visual field because there are too many kinds of conjunctions. There are a million visual locations, so the number of processors needed would be a million multiplied by the number of possible logical conjunctions.

The number of colours that we can discriminate times the number of contours times the number of depths times the number of directions of motion times the number of velocities and so on is an astronomical number.

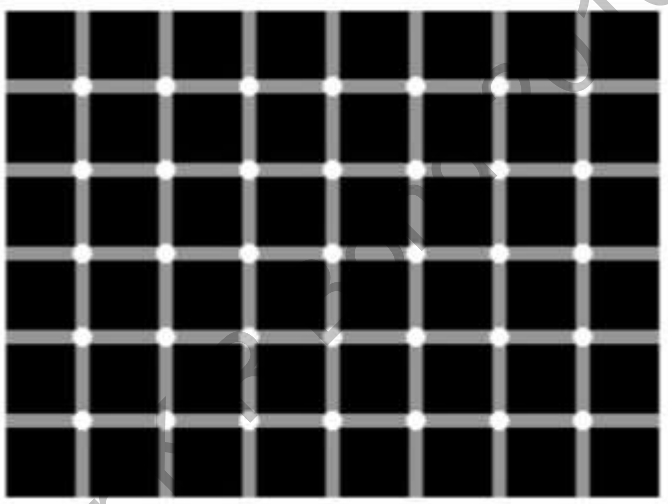
Conjunctions are combinatorial



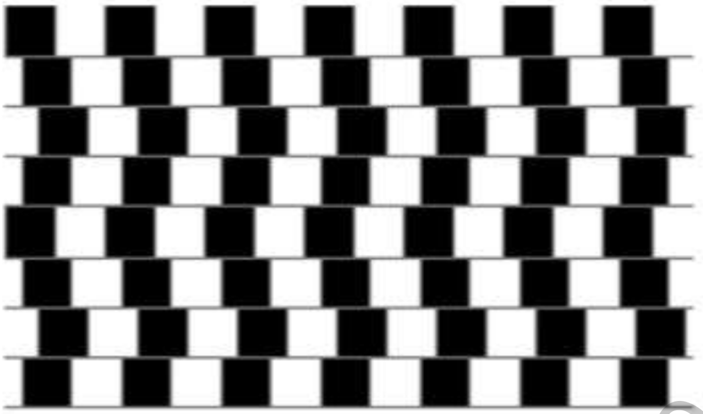
The conjunctions are detected only by a programmable logic machine that looks at one part of the visual field at a time through a narrow moveable window and passes on its answer to the rest of cognition.

If the conscious processor is focussed at one location, the features at other locations should float around unglued. A person not deliberately attending to a region should not know whether it contains a red X and a green O or a green X and a red O. The colour and the shape should float in separate planes and the conscious processor brings them together at a particular spot.

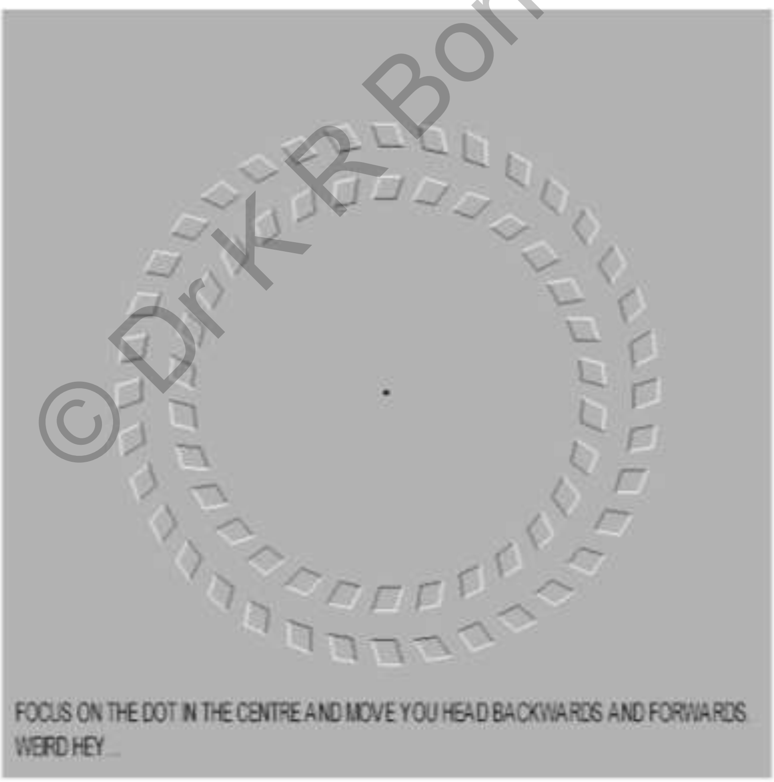
Optical illusions

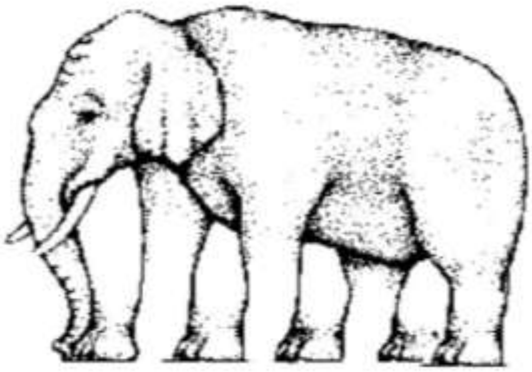


Count the black dots! :o)



Are the horizontal lines parallel or do they slope?





How many legs does this elephant have?

Biology

This topic is about biology-inspired computing.

You will need a copy of the programming environment Greenfoot or StarLogo TNG to investigate ant-colonies or a similar simulation system such as programming environment Processing.

Read the following article.

Social insects and self-organization

An ant is quite a simple animal. Its behavioural repertory is limited to ten to forty elementary behaviours. Yet, anthills are very complex. One can find nursery, warehouses or kitchen gardens. Some individuals forage, others take care of the eggs, repair the nest or protect the anthill against miscellaneous threats. What is the secret? How can such mindless animals achieve such complex organization?
Division of labour could be the key. Ants are highly specialized, so specialized that some individuals have to be fed by others; they are unable to get food by themselves. In economics, division of labour leads to efficiency, but, to function properly, some sort of supervision is necessary. The different tasks have to be coordinated. Yet no such supervisors exist in anthills. No ants (and particularly not the queen) are able to manage this exploit. Nevertheless, the coordination necessarily exists; it results from some sort of **self-organization process.**

Let us examine foraging strategies in ants to illustrate this idea.

At the beginning, a number of ants are walking, more or less randomly, outside the nest. They are looking for food. All along their way, they deposit a light trail of pheromones. When an ant finds some food, it returns home, depositing a stronger trail (the intensity of the trail possibly depends on the richness of the discovered resource). Since ants have trail-following behaviour, a growing number of individuals will tend to follow it and to reach the food. When they return, they reinforce the trail. Positive feedback (self-amplification) therefore occurs. More individuals reinforce the trail, attracting new individuals, who in their turn reinforce the trail...

In this example, the ants don't communicate directly. Information is exchanged through modifications of the environment (here local gradients of pheromones). This type of communication is known as stigmergy. This concept was proposed by P.P. Grassé in 1959. Studying nest reconstruction in termites, Grassé showed that it doesn't rely on direct communication between individuals. The nest structure itself coordinates the workers' tasks, essentially through local pheromone concentrations. The state of the nest structure triggers some behaviours, which then modify the nest structure and trigger new behaviours until the construction is over. The process is similar in ant foraging.
The ants tend to follow pheromone trails, but it is only a tendency. There is at any time a positive probability that an ant will abandon the trail and move more or less randomly. It is then possible that the "lost" ant could find a new resource, perhaps far richer than the one that

was previously exploited. By constructing a new trail, this ant will attract new individuals and a new positive feedback loop will be set up.
Finally, when satiety occurs, or when the resource is used up, a negative feedback loop occurs. For example, if pheromone decay is quick enough, when the resource is exhausted fewer and fewer ants will tend to follow the trail and it will progressively disappear.

Self-organization in social insects is interpreted through four main mechanisms[8]:

1. The existence of multiple interactions.
2. Amplification through positive feedback.
3. Negative feedback.
4. Amplification of fluctuations. In the previous example, the fluctuation is when an ant abandons the pheromone trail; it is amplified by the positive feedback loop which then occurs.

Ants foraging process in some species have been analyzed by J.-L. Deneubourg[9] (Université Libre de Bruxelles). He notably showed how ants can find the best (shortest) way to reach a resource. In a nutshell, the accumulation of pheromones is faster on the shortest route, so positive feedback therefore gives it priority.
On this basis, F. Moyson and B. Manderick[10], followed by M. Dorigo[11] proposed the concept of
"Ant Colony Optimization" (ACO). Dorigo applied this process to the travelling salesman problem and then extended it to a whole class of optimization problems. Such algorithms can now be found in telecommunications routing, the design of electronic circuits or -- for example -- the organization of industrial processes.

Jean-Philippe Rennard 02/2003
http://www.rennard.org/alife

8 Bonabeau E., Dorigo M., Théraulaz G.,Swarm Intelligence. From Natural to Artificial Systems, Oxford University Press, 1999, p. 8-14.

9 See for example : Deneubourg J.-L. et al., « Plan d'organisation et population dans les sociétés d'insectes », p. 141-155, dans Prigogine I. (dir.), L'homme devant l'incertain, Paris, Odile Jacob, 2001.

10 Moyson F. et Manderick B., 1988. « The Collective Behaviour of Ants: an Example of Self-Organisation in Massive Parallelism », Proceedings of the AAAI Spring Symposium on Parallel Models of Intelligence. Stanford, California.

11 Dorigo M., Gambardella L.M., Ant Colonies for the Travelling Salesman problem, BioSystems, 43, 1997.

Closed and Open Computing

This topic is about the mathematical worldview of computing, i.e. algorithms and the interactive worldview of computing i.e.

Closed-box Computation

The theory of computation views computing as a *closed-box* transformation of inputs to outputs, completely captured by Turing Machines. This view predates the establishment of computer science as a discipline, having been part of mathematics before the 1960s.

This view assumes that all computation is closed, i.e. that there is no input or output taking place during computation; any information needed during the computation is provided at the outset as part of the input.

This **mathematical worldview** of computing can be summarised as follows:

All computable problems are function-based

This does not seem unreasonable a viewpoint to take, after all mathematics has been used as a foundation of physics and other scientific disciplines and was the viewpoint adopted by computer scientists in the 1950s and 1960s. Turing Machines (TM) transform input strings into output strings. Therefore, TMs have served as a formal model for function-based computing.

What is a Function?

A function is like a machine: it has an input and an output and the output is related somehow to the input.

The classic way of writing a function is "**f(x) = ...** " but there are other ways.

Domain, Range and Codomain

In its simplest form the domain is all the values that go into a function and the range is all the values that come out.

Functions

A function ***relates*** an input to an output:

Example: this tree grows 30 cm every year, so the height of the tree is related to its age using the function h:

h(age) = age × 30

So, if the age is 10 years, the height is h(10) = 300 cm

Saying "h(10) = 300" is like saying 10 is somehow related to 300 or 10 → 300

Input and Output

But not all values may work!

- The function may not work if you give it the wrong values (such as a negative age),
- And knowing the values that can come out (such as always positive) can also help

So we should really say all the values that **can go into** and **come out of** a function.

This is best done using **Sets ...**

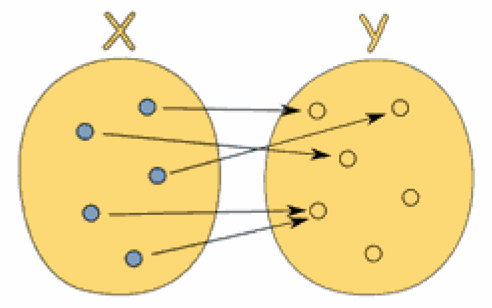
A set is a collection of things, such as numbers.

Here are some examples:

Set of even numbers: {..., -4, -2, 0, 2, 4, ...}
Set of odd numbers: {..., -3, -1, 1, 3, ...}
Set of prime numbers: {2, 3, 5, 7, 11, 13, 17, ...}
Positive multiples of 3 that are less than 10: {3, 6, 9}

In fact, a function is defined in terms of sets:

A function relates each element of a set with exactly one element of another set (possibly the same set).

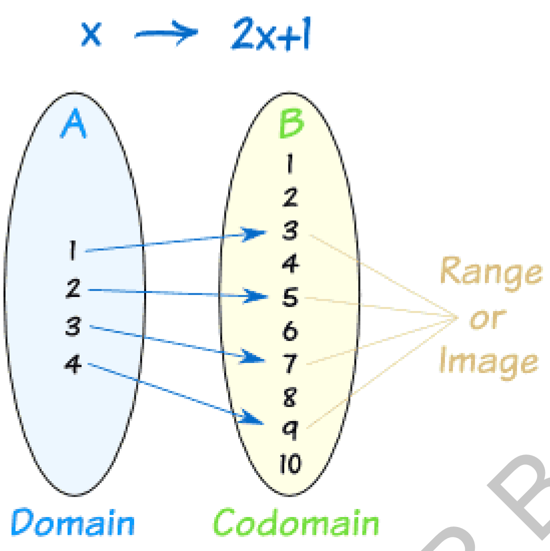


Domain, Codomain and Range

There are special names for what can go into, and what can come out of a function:

- What can go into a function is called the Domain
- What may possibly come out of a function is called the Codomain
- What actually comes out of a function is called the Range

Let us look at a simple example:



In this illustration:

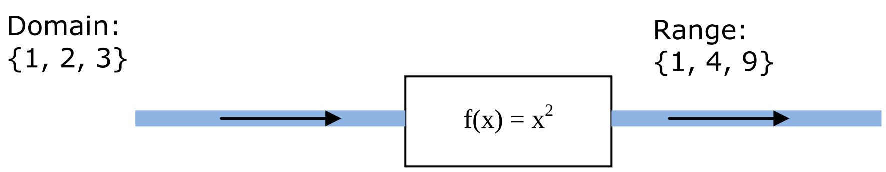
- the set "A" is the Domain,
- the set "B" is the Codomain,
- and the set of elements that get pointed to in B (the actual values produced by the function) are the Range, also called the Image.

Now, what comes out *(the Range)* depends on what you put in *(the Domain)* ...

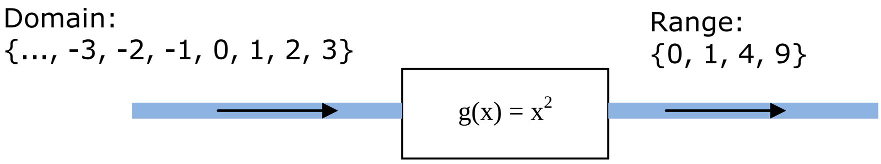
... but **YOU** can define the Domain!

In fact the Domain is an essential part of the function. A different Domain and you have a different function.

Example: a simple function like $f(x) = x^2$ can have the **domain** (what goes in) of just the counting numbers {1,2,3,...}, and the **range** will therefore be the set {1,4,9,...}



And another function $g(x) = x^2$ can have the domain of integers {...,-3,-2,-1,0,1,2,3,...}, in which case the range will be the set {0,1,4,9,...}



Even though both functions take the input and square it, they operate on a **different set of inputs**, and so give a different set of outputs.

Also they will have different properties.

For example f(x) always gives a unique answer, but g(x) can give the same answer with two different inputs (such as **g(-2)=4**, and also **g(2)=4**)

So, the domain is an essential part of the function.

Does Every Function Have a Domain?

Yes, but in simpler mathematics you never notice this, because the domain is assumed:

- Usually it is assumed to be something like "all numbers that would work".
- Or if you are studying whole numbers, the domain is assumed to be whole numbers.
- etc.

But in more advanced work you need to be more careful!

Codomain vs Range

The Codomain and Range are both on the output side, but are subtly different.

The Codomain is the set of values that could possibly come out. The Codomain is actually part of the definition of the function.

The Range is the set of values that actually do come out.

Example: you can define a function f(x)=2x with a domain and codomain of integers (because you say so).

But by thinking about it you can see that the range (actual output values) would be just the even integers.

So the codomain is integers (you defined it that way), but the range is even integers.

The Range is a subset of the Codomain.

Why both? Well, sometimes you don't know the exact range (because the function may be complicated or not fully known), but you know the set it lies in (such as integers or reals). So you define the codomain and continue on.

The Importance of Codomain

Question: Is ***square root*** a function?

If you say the codomain (the possible outputs) is **the set of real numbers**, then square root is **not a function**! ... is that a surprise?

The reason is that there could be two answers for one input, for example ***f(9) = 3*** or ***-3***

A function must be single valued. It cannot give back two or more results for the same input. So "f(3) = 9 or 11" is not right!

However, it can be fixed by simply **limiting the codomain** to non-negative real numbers.

In fact, √, the radical symbol (e.g. √4) always means the principal (positive) square root, so √x is a function because its codomain is correct.

So, **what you choose for the codomain** can actually affect whether something is a **function or not**.

Notation

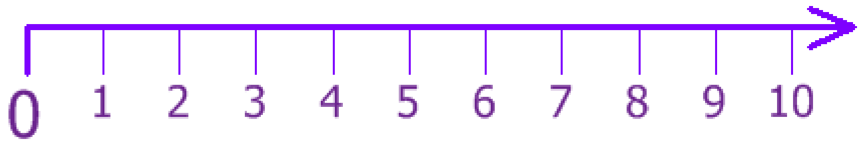
Mathematicians don't like writing lots of words when a few symbols will do. So there are ways of saying "the domain is", "the codomain is", etc.

The following is a neat way:

$f: \mathbb{N} \rightarrow \mathbb{N}$	this says that the function "***f***" has a domain of "ℕ" (the natural numbers), and a codomain of "ℕ" also.
$f: x \mapsto x^2$ *or* $f(x) = x^2$	and either of these say that the function "f" takes in "x" and returns "x^2"

Whole Numbers

Whole Numbers are simply the numbers 0, 1, 2, 3, 4, 5, … (and so on)



No Fractions!

Counting Numbers

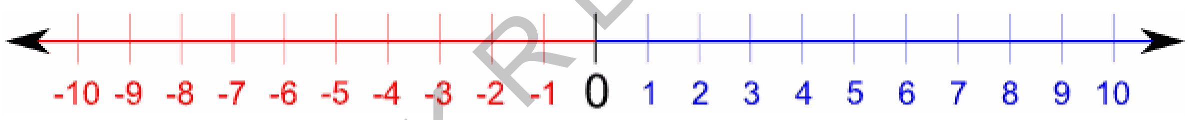
Counting Numbers are Whole Numbers, but **without the zero** because you can't "count" zero. So they are 1, 2, 3, 4, 5, … (and so on).

Natural Numbers

"Natural Numbers" can mean either "Counting Numbers" (1, 2, 3, etc), or "Whole Numbers" (0, 1, 2, 3, etc), there is disagreement on the definition.

Integers

Integers are like whole numbers, but they **also include negative numbers** ... but still no fractions allowed!



So, integers can be negative {-1, -2,-3, -4, -5, … }, positive {1, 2, 3, 4, 5, … }, and zero {0}

Confusing

Just to be confusing, *some* people say that whole numbers can also be negative, so that would make them exactly the same as integers and sometimes people say that zero is NOT a whole number.

Notation

$\mathbb{N}$ denotes to the set of natural numbers = {0, 1, 2, 3, 4,….}

$\mathbb{Z}$ denotes the set of integers = {…., -2, -1, 0, 1, 2, 3, …}

$\mathbb{Z}^+$ is the set of positive integers = {1, 2, 3, 4, 5, …}

$\mathbb{Q}$ is the set of rational numbers = $\{p/q \mid p \in \mathbb{Z}, q \in \mathbb{Z} \text{ and } q \neq 0\}$

$\mathbb{R}$ is the set of real numbers

Data Type

In computer science the concept of data type is built upon the concept of a set. In particular, data type is the name of a set together with a set of operations that can be performed on objects from that set. For example *Boolean* is the name of the set {0, 1} together with operators on one or more of elements of this set, such as AND, OR and NOT.

Let f be the function that assigns the last two bits of a bit string of length 2 or greater to that string. For example, f(11010) = 10. Then the domain of f is the set of all bit strings of length 2 or greater and both the codomain and the range are the set {00, 01, 10, 11}.

Let $f: \mathbb{Z} \rightarrow \mathbb{Z}$ assign the square of an integer to this integer. Then $f(x) = x^2$, where the domain of *f* is the set of all integers, we take the codomain of *f* to be the set of all integers and the range of *f* is the set of all integers that are perfect squares, namely {0, 1, 4, 9, ..}

Domains and Codomains in programming Languages

The domain and codomain of functions are often specified in programming languages.

For instance, the Pascal statement

Function Trunc(x : Real) : Integer;

states that the domain of the **Trunc** function is the set of real numbers and its codomain is the set of integers.

Factorial function

$$f : \mathbb{N} \rightarrow \mathbb{Z}^+ \text{ denoted by } n!.$$

The value of $f(n) = n!$ is the product of the first n positive integers, so $f(n) = 1 \cdot 2 \cdot 3 \cdot (n - 1) \cdot n$ and $f(0) = 0! = 1$

We have

$f(1) = 1! = 1$,

$f(2) = 2! = 1 \cdot 2 = 2$,

$f(6) = 6! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720$ and

$f(20) = 20! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11 \cdot 12 \cdot 13 \cdot 14 \cdot 15 \cdot 16 \cdot 17 \cdot 18 \cdot 19 \cdot 20 =$ 2,432,902,008,176,640,000

Exercises

1. Write a function that assigns to each pair of positive integers the maximum of these two integers.
2. Write a function that assigns to each natural number belonging to the subset {0, 1, …, 10} the factorial of this natural number.
3. Write the function that assigns to a bit string the numerical position of the first 1 in the string and that assigns the value 0 to a bit string consisting of all 0s.

Algorithm

The notion of an algorithm precedes computer science having been the concern of mathematicians for centuries. It can be dated back to a 9th century treatise by a Muslim mathematician Al-Koarizmi, after whom algorithms were named.

Algorithms originated in mathematics as "recipes" for carrying out function-based computation, that can be followed mechanically. Like mathematical formulae, algorithms tell us how to compute a value. Unlike them, algorithms may involve *loops* and *branches*.

Given some finite input *x*, an algorithm describes the steps for effectively transforming it to an output *y*, where *y* is *f(x)* for some recursive function *f*.

Algorithmic Computation

Algorithmic computation, on the other hand, is computation performed in a closed-box fashion, transforming a finite input, determined by the start of the computation, to a finite output, available at the end of the computation, in a finite amount of time. **No new input is accepted once the computation has begun**.

Open-box Computation

Web services (e.g. web server), intelligent agents (agent = anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors), operating systems and graphical user interfaces are interactive applications rather than algorithmic ones. Their job is to provide ongoing services over time.

According to the interactive view of computation, interaction (communication with the outside world) happens *during* computation, not before or after it. Hence, computation is an ongoing process rather than a function-based transformation of an input to an output. This view of computation is not modelled by Turing machines, which capture only the computation of functions.

The theory of computation needs to be extended from functions and algorithms to processes and services.

Interactive Worldview

In this worldview computation is viewed as an ongoing process that transforms inputs to outputs – e.g., control systems, operating systems.

The nature of operating systems and other interactive systems is that these systems never terminate and therefore formally never produce an output. Yet they do compute and their computation is useful.

Clearly, the mathematical worldview no longer reflects the nature of computational problems. An example of such a problem is *driving home from work* (Wegner, 1997[12]; Eberbach, Goldin and Wegner, 2004[13]):

Driving home from work: *create an automatic car to drive us home from work, where the locations of both work and home are provided as input parameters. No interaction with its environment is allowed once processing begins, i.e. it relies upon the mathematical worldview*

Input to this problem: Detailed map, current weather report, etc (all encoded as a finite string)

Output of this problem: Two time series –

1. One that shows position of the steering wheel (e.g. in degrees from the vertical) during the drive
2. Another to show the force on the accelerator and brake pedals during the drive

Assuming that the driving is to take place in a real-world environment, this problem is not computable within a function-based computational paradigm, i.e. one that does not allow interaction with its environment..

Why?

---

[12] Peter Wegner. Why Interaction is More Powerful Than Algorithms. Comm. ACM, May 1997

[13] Eugen Eberbach, Dina Goldin, Peter Wegner. Turing's Ideas and Models of Computation. In Alan Turing: Life and Legacy of a Great Thinker, ed. Christof Teuscher, Springer 2004.

Consider the input to such a function.

It must be detailed enough so the car can predict the direction and speed of the wind at every moment of the drive, so as to compensate for it.

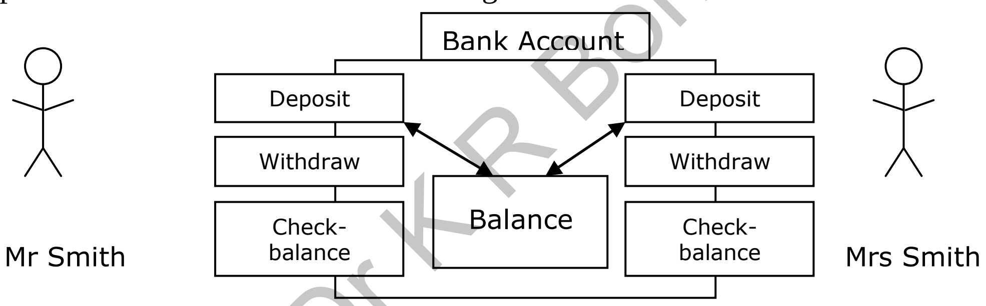
It should also anticipate the location of all pedestrians, so as to avoid running over them.

This is impossible, there is no computable function that will determine, given some amount of a priori (knowledge independent of experience) information, all the real world factors that are necessary to ensure the car's safe arrival at its destination. An assertion to the contrary would endow Turing Machine's with a power to predict the future that is tantamount to a resolution of the predestination debate.

However, the problem of driving home from work is computable – by a control mechanism, as in a robotic car, that continuously receives video input of the road and actuates the steering wheel and brake accordingly. This computation, just as that of operating systems, is interactive where input and output happen *during* the computation, not before or after it.

In the interactive world view *entanglement* of inputs and outputs, where later inputs to the computation depend on earlier outputs and vice versa, are allowed. Such entanglement is impossible in the mathematical worldview, where all inputs precede computation and all outputs follow it. In the mathematical worldview the computation is independent of experience.

Consider a shared bank account belonging to a married couple and two geographically separated cash machines as shown in Figure ?



Let's say that the balance of the account starts out at £2000 and changes To £1500 then £1700 then £1400 then £1000.

What changes have taken place?

£500 is withdrawn

£200 is deposited

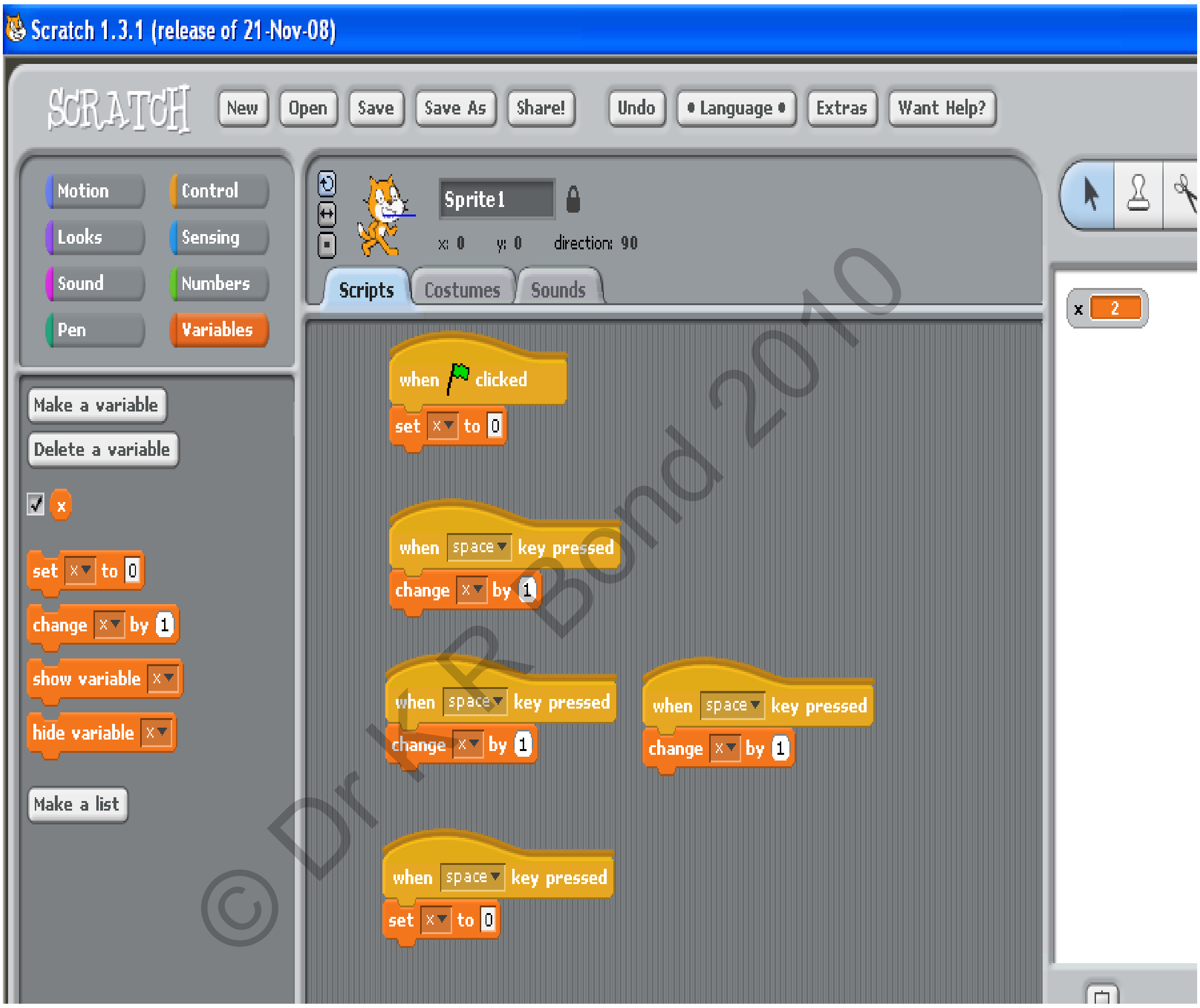
£300 is withdrawn

£400 is withdrawn

The list above says nothing about who performed these operations – deposit and withdraw - but if only Mr and Mrs Smith have access to the account then the deposit/withdraw operations will have been carried out by Mr Smith and/or Mrs Smith. If there is another operation called check-balance then the result returned depends upon whether the operations deposit and withdraw have been carried out. Thus the effect of a check-balance operation of a bank account is not uniquely determined by the operation alone, since it depends on changes of state by deposit and withdraw operations that cannot be predicted or controlled. An object’s operations return results that depend on changes of state controlled by unpredictable actions. Operations are therefore not algorithmic.

Scratch Example

In the Scratch example shown in Figure ? what is value of x after the program is started and the space bar is pressed?



The answer is unpredictable. If we edit the program by changing the first key pressed command to respond to when the letter “a” is pressed, the second to when the letter “b” is pressed, the third “c” and the fourth “d”. The answer depends on the history of key presses. The answer is nonalgorithmic.

This is quite typical of the interaction paradigm. Objects are interactive agents. Objects remember their past and interact through an interface of operations that share a hidden state.

Programming in the large is inherently interactive and cannot be expressed by or reduced to programming in the small. The behaviour of airline reservation systems and other embedded systems cannot be expressed by algorithms.

From Turing to Interaction Machines[14]

Turing showed in the 1930s that algorithms in any programming language have the same transformation power as Turing Machines, computing precisely *computable functions*. This precise characterization of what can be computed established the respectability of computer science as a discipline. However, the inability to compute more than the computable functions by adding new primitives proved so frustrating that it was called the Turing tar pit.

Turing machines transform strings of input symbols on a tape into output strings by sequences of state transitions - http://aturingmachine.com/index.php. Each step reads a symbol from the tape, performs a state transition, writes a symbol on the tape and moves the reading head. Turing machines cannot accept external input while they compute: they shut out the external world and therefore cannot model the passage of external time.

The hypothesis that the formal notion of computability by Turing machines corresponds to the intuitive notion of what is computable, known as Church's thesis has been accepted as obviously true for 50 years. However, when the intuitive notion of what is computable is broadened to include interaction.

Turing machines extended by adding *input* and *output* actions that support dynamic interaction with an external environment are called *interaction machines*. Interaction machines transform closed-box computing systems into open systems. Interaction machines cannot be modelled by algorithms. The actual time of execution becomes crucial in an interaction machine because interaction machines depend on their interaction histories. Algorithmic time on the other hand is measured not by the actual time of execution but by the number of instructions executed in order to provide a hardware-independent measure of logical complexity.

Open systems are defined as systems that can be modelled by interaction machines, while closed systems are modelled by algorithms.

Interfaces as Behaviour Specifications

The negative result that interaction is not expressible by algorithms leads to positive new approaches to system modelling in terms of interfaces. This means specifying the system's parts and its forms of behaviour rather than the complete system. A system satisfies its requirements if it supports specified modes of use even though correct behaviour for a given mode of use is not guaranteed and complete system behaviour for all possible modes of use cannot be specified. Result checking is needed during execution to verify that results actually obtained are valid. Results checking for tasks such as driving with visual feedback is performed automatically by people.

[14] The Paradigm Shift from Algorithms to Interaction, Wegner, Brown University, 1996

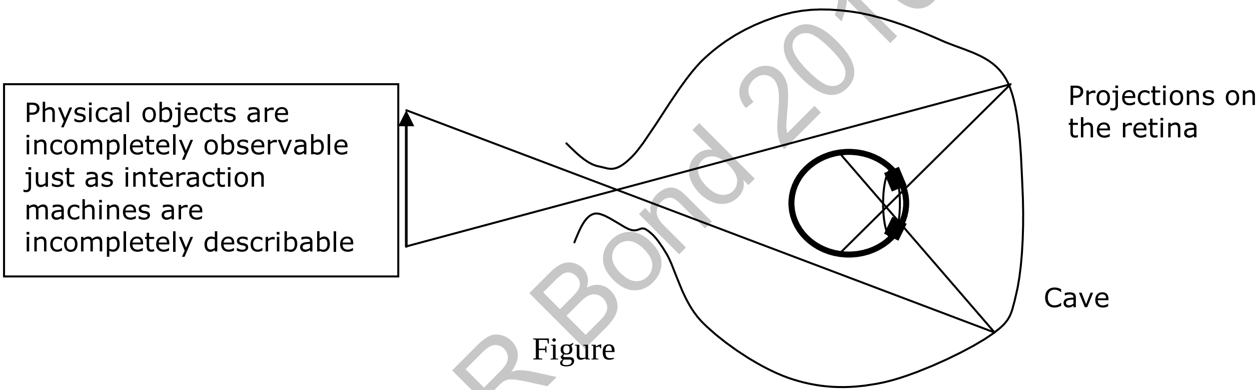
Philosophy

From Rationalism to Empiricism

The previous paragraphs have attempted to show the limits of logic applied to interactive systems. Just as Gödel, Turing and Church argued that logic cannot completely prove all mathematical theorems (Godel discovered that the integers cannot be completely described by logic, demonstrating the limitations of formalism in mathematics) so it is not possible to achieve the goal of proving correctness for interactive systems.

Rationalism

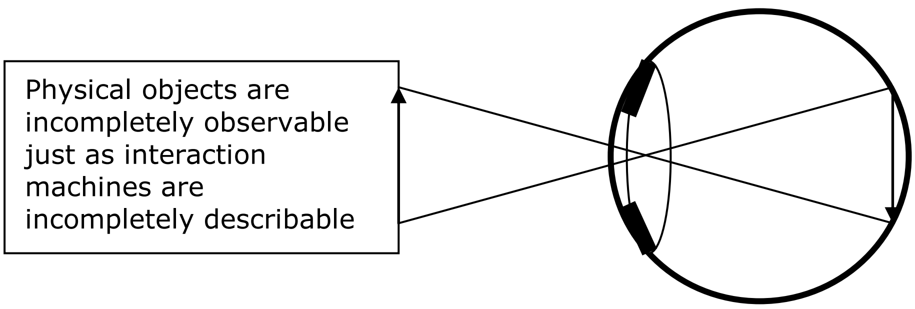
Plato concluded that abstract ideas are more perfect and therefore more real than physical objects like chairs and tables. Plato used the parable of the cave, which compares humans to cave dwellers who can observe only shadows of reality on the walls of their cave but not actual objects in the outside world, to argue that observation cannot completely specify the inner structure or behaviour of observed objects. Projections of light on our retinas serve as incomplete cues for constructing our world of solid tables and chairs – Figure ?.



Figure

Empiricism

Plato's scepticism concerning empirical science contributed to the 2000-year hiatus in the evolution of empiricism. Empiricists accept the view that perceptions are reflections of reality but disagree with Plato on the nature of reality, believing that the physical outside the cave is "real" but unknowable. Fortunately, "complete" knowledge is unnecessary for empirical models of physics, because they achieve their pragmatic goals of prediction and control by dealing entirely with observable reflections. Figure ? shows a schematic of the interface between the real world and the brain, i.e. an image projected onto the retina.



Figure

Modern empirical science rejects Plato's belief that incomplete knowledge is worthless, using partial descriptions (shadows) to control, predict and understand the objects that shadows represent. Equations capture quantitative properties of phenomena that they model without requiring a complete description. Similarly, computing systems can be specified by interfaces that describe properties judged to be relevant while ignoring properties judged irrelevant. Plato's cave, properly interpreted, is a metaphor for empirical abstraction in both natural science and computer science.

Plato, Descartes, Turing Machines, Hume and Kant

Turing machines correspond to Platonic ideals in focusing on mathematical at the expense of empirical models. To realize logical completeness, they sacrifice the ability to model external interaction and real time. The extension from Turing to interaction machines and from procedure-oriented to object-based systems, is the computational analogue of liberation from the Platonic world to the world of empirical science. Interaction machines free computing from the "Turing tar pit" of algorithmic computation, providing a conceptual model for software engineering, AI agents (agent = anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors), and the real (physical) world.

Descartes' (31st March 1596 – 11th February 1650 also known as Renatus Cartesius, French philosopher, mathematician, physicist and writer) quest for certainty led to the rationalist credo "cogito ergo sum", "I think therefore I am". The rationalist credo succinctly asserts that thinking is the basis of existence and implies that "certain" knowledge of the physical world is possible only through inner processes of algorithmic thinking. Hume on the other hand was called an empiricist because he showed that inductive inference and causality are not deductive and that rationalist models of the world are inadequate. Kant, "roused from his dogmatic slumbers" by Hume, wrote the *Critique of Reason* to show that "pure" reasoning about necessarily true knowledge was inadequate to express contingently true knowledge of the real world.

Social and scientific rationalism have common roots in the deep-seated human desire for certainty. Empiricism has displaced rationalism in the sciences. Turing machines reflect rationalist reasoning paradigms of logic rather than empirical paradigms of physics. Algorithms and Turing machines, like Cartesian thinkers (followers of the thinking of Renatus Cartesius), shut out the world during the process of problem solving. Abstraction is a key tool in problem solving because it enables systems to be simplified. Incompleteness is the key distinguishing mechanism between rationalist, algorithmic abstraction and empiricist, interactive abstraction. Incomplete behaviour is the essential ingredient that distinguishes both interactive from algorithmic models of computing and empirical from rationalist models of the physical world.

Computer Science is a Science

Irreducibility of interaction to algorithms enhances the intellectual legitimacy of computer science as a discipline distinct from mathematics and by clarifying the nature of empirical models of computation, provides a technical rationale for calling computer science a science. Interfaces of computing systems are the computational analogue of shadows on the walls of Plato's cave, providing a framework for system description that is more expressive than algorithms and captures the essence of empirical computer science.

Systems

(Adapted from Interactive Computation, The New Paradigm, Goldin, Smolka, Wegner (Eds), ISBN 3-540-34666-X, 2006, “Interaction, Computation and Education”, Lynn Andrea Stein, p463-p483) – need to seek permission to reproduce.

Closed Systems

The calculation model of computation is really quite different from many of our everyday experience. In the calculation metaphor, the outside world doesn’t really have a role to play in the sequence of steps that constitutes a computation.

Open Systems – reactive systems

In open systems, input is continually arriving and output is continually being produced.

Consider the task of organising the operation of a cafe or restaurant.

Problem: How to serve customers on an ongoing basis.

(The Web, the modern word processor and an operating system provide services on an ongoing basis)

Constraints: The restaurant services must be provided simultaneously – it won’t do to wait for the first customer to finish before taking the second’s order.

Input will not arrive at the beginning; instead, customers will continually walk in the door.

Output is not what is done just before the restaurant closes; it is a steady production.

Input: What is monitored.

Output: What the restaurant does

Initial thoughts:

The restaurant is a community of interacting entities (things).

Who are the members of the community?

1. Waiting staff
2. Kitchen staff
3. Business staff

How should they interact?

(**Key notions here are interface and protocol**: An interface is the interactive equivalent of a functional description, specifying what an entity requires, what it produces, what behavioural contracts the entity can be expected to subscribe to. A protocol describes the precise choreography of an interaction, including what each party does in what temporal sequence and how information moves back and forth).

One possible way:

Data-structure based protocol for the waiter to communicate orders on a piece of paper, then hang the piece of paper in the kitchen staff’s window. The state of the paper (including its physical location) serves as a cue to the kitchen staff as to what food preparation remains on the order. When the food is delivered to the waiter, the scrap of paper is thrown away.

Decomposition: Once we design the community and its interactions we need to apply the traditional techniques of recursive decomposition. Of each entity, we ask how it is made. For example, the waiting staff might consist of the maitre d'hotel, one or more serving waiters and staff to clear the tables. Among them each has a distinct set of responsibilities and certain protocols for interaction. From the kitchen's perspective, the waiting staff may be approximated as a single entity that periodically delivers an order request and retrieves platters of food but from among the waiting staff the subdivision of community is visible.

Stepping outside of the restaurant, we see that the restaurant is itself embedded in a community. That community involves the restaurant's suppliers, the tax collector, the landlord and many others. The same model and questions – Who are the members? How do they interact? What's inside each? – apply.

The restaurant model is composed of ongoing persistent autonomously active entities: the staff. They are coupled together using various interaction protocols. The entire system is evaluated based on ongoing behaviour, rather than any end result. Computation today is like running a restaurant.

Bin Sort

Given: A collection of differently coloured balls

Task: Sort these balls into baskets, one corresponding to each colour represented.

Sequential Bin Sort

1. Pick up a ball from the input bucket
2. Consider the first basket
3. If colour of ball matches colour of current basket, put ball into basket and go to step 1
4. Otherwise (the colour does not match) consider the next basket
5. Go to step 3

This program can be executed by a single thread of computation. It will eventually wind up placing each ball in the appropriately coloured basket.

This is a conventional algorithmic computation:

- Functional specification matches the requirements of the problem
- All input is available at beginning of the problem
- The computation's result is its final state

Complexity analysis: Task will be completed on average in ***nk/2*** iterations (where n is the number of balls and k the number of baskets) and at worse will require ***nk*** iterations.

Community Bin Sort 1

Instead of breaking the problem into steps, we break it into entities – one for each basket (or colour). Next we arrange these entities in a line, with buffers between them. Each basket is associated with two buffers: an input buffer and an output buffer. We start with all the coloured balls in the input buffer of the red basket (contents of bucket of coloured balls feeds into red basket's input buffer). The output buffer of the red basket is connected to the input buffer of the next basket, the yellow basket, and so on. A coloured ball placed in an output buffer is immediately transferred to the input buffer to which it is connected.

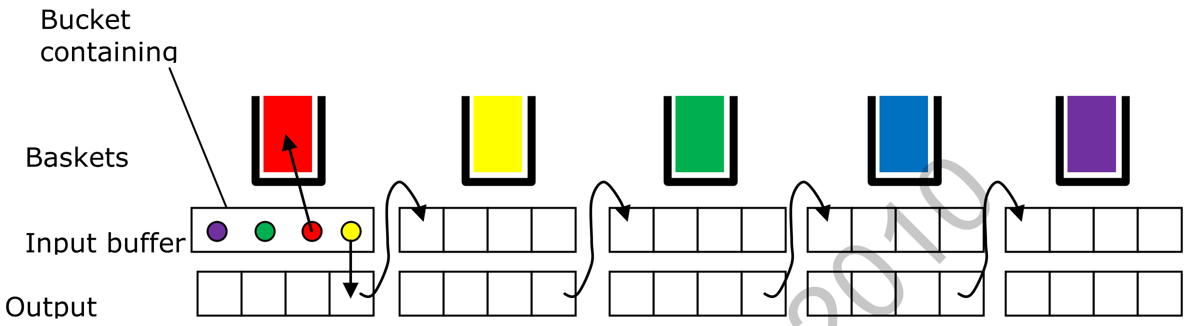


Figure ? Community bin sort 1

Each entity follows the interactive rule:

1. Pick up a ball from your input buffer
2. If the colour of the ball matches the colour of your basket, put the ball into your basket
3. Otherwise (it is not your colour), put it into your output buffer

    (Go back to 1)

The "Go back to 1" control flow is implicit because the default behaviour for an interactive entity is to keep processing – to keep looping over its inputs forever.

Now we have a system that consists of entities working in parallel. This system differs from the sequential one in that it contains multiple active entities – multiple threads of control – and simultaneous activity.

Community Bin Sort 2

In Community Bin Sort 1 one of the entities has privileged access to the original bucket of balls. There is no particular reason why this should be the case. Instead, the "program" can be changed so that each entity uses the original bucket as its input buffer and as its output buffer. Note that there has been a change of topology of the community, i.e. its interconnections. The new topology involves only one shared bucket. In this configuration each entity picks up a ball from the (single) shared central bucket, keeps it if it matches, and otherwise returns it to that bucket.

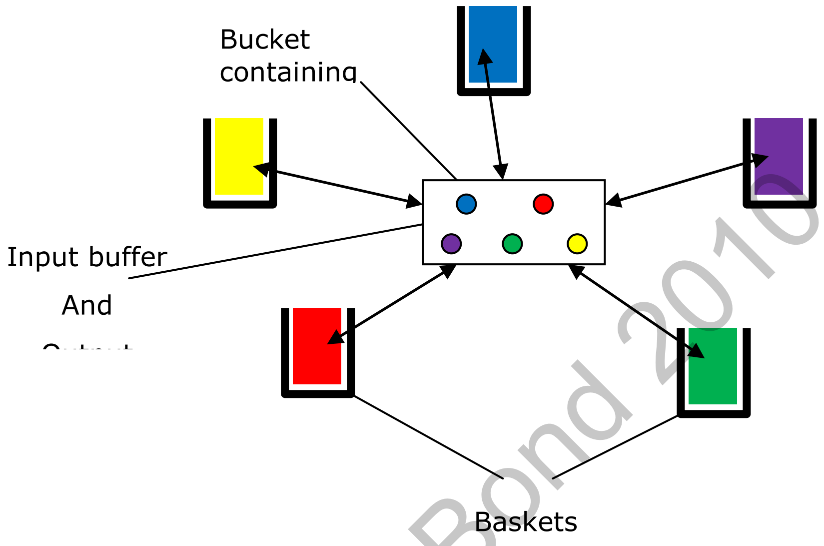


Figure ? Community bin sort 2

Thus each entity follows the interactive rule:

1. Pick up a ball from your input buffer
2. If it is your colour, put it in your basket
3. Otherwise (it is not your colour), put it back into your input buffer

   (Go back to 1)

The "Go back to 1" control flow is implicit because the default behaviour for an interactive entity is to keep processing – to keep looping over its inputs forever.

This system is not guaranteed to complete. For example the red-basket entity might continually pick up and put down a blue ball, preventing it from ever reaching the blue basket.

The sort is a kind of randomized sort.

Community Bin Sort 2 Combined

We can take the output baskets – red, green, blue, yellow, purple – of the first level sort and use each as the input to the next level of sorting. The first level of entities sorts by major colour family, the second by shading within that colour family, and so on. Given the way that the problem has been decomposed into a meshing of coarse-grained through to increasingly finer and finer grained sorts the problem solution scales easily.

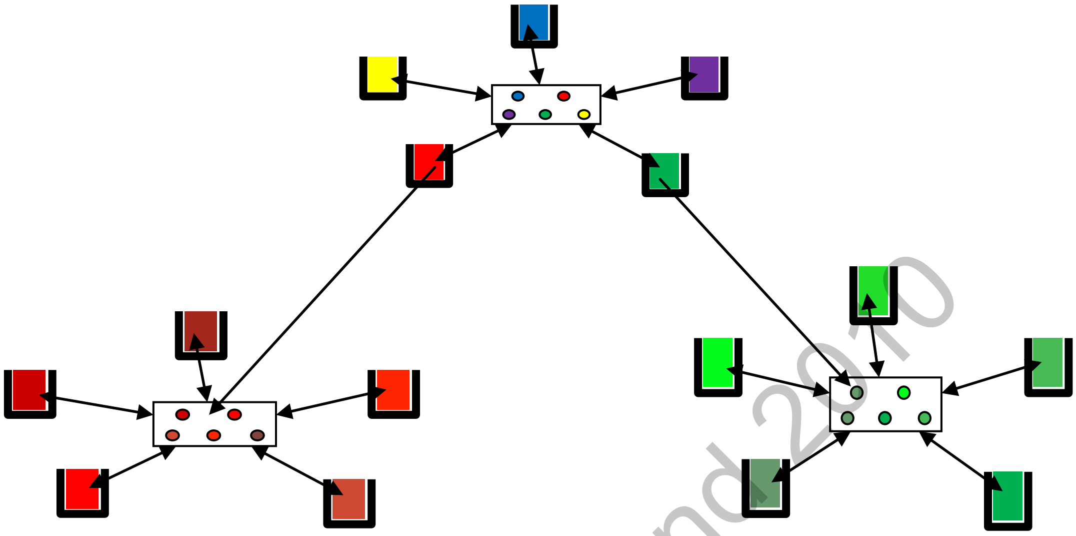


Figure ? Hierarchical community sorting

Network Routing

Instead of using a fixed bucket of coloured balls as input we supply coloured balls continuously over a conveyor belt. The interactive sort is now on-line, it receives input continuously. Now, there is no end to the supply and therefore no final state by which to judge the performance of our computational community.

Instead we ask questions about more engineering terms:

latency and throughput, correctness and completeness.

How long does each ball spend travelling through the system?

Are ordering constraints preserved?

Is it guaranteed to eventually find its home?

Now imagine replacing coloured balls by message packets. It is message packets that need to be sorted, i.e. routed to the correct basket where now a basket is the computer (network node/station) that the message packet is addressed to.

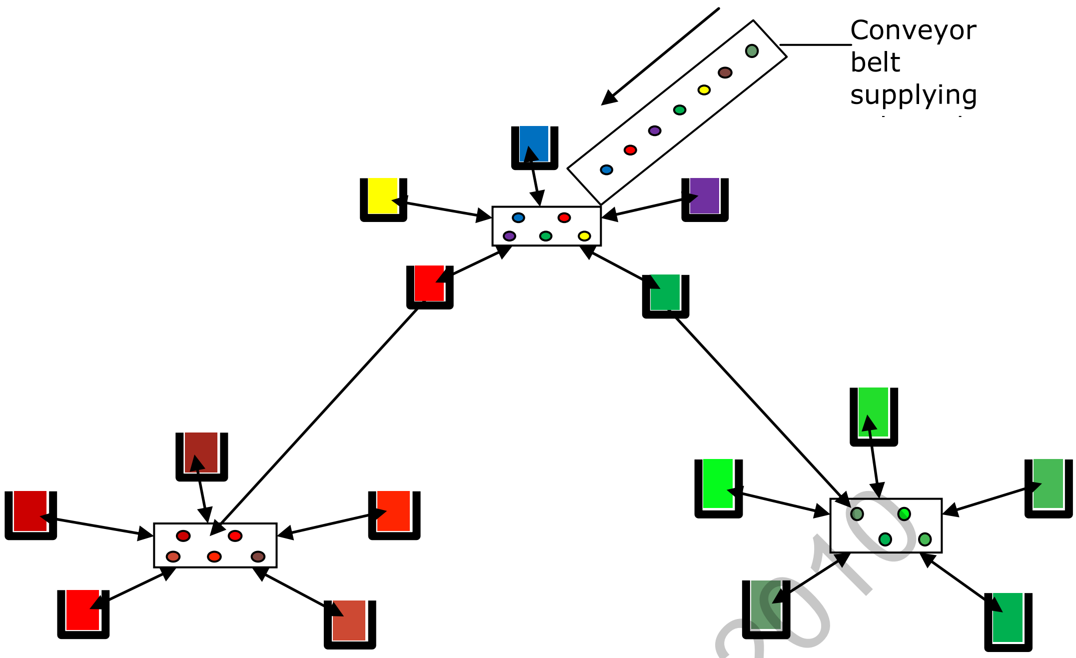


Figure ? On-line hierarchical community sort – network routing

The final version of this program is a simplified form of the programs that run our worldwide communication networks. The program consists of the interaction of a community of interacting agents.

From Simple Infinite Echo Loop to Networked Video Games

The bin sorting/network routing example illustrates the way in which a traditionally sequential problem can be completely transformed by recasting it in an interactive framework.

Simplest Program

We begin with the simplest interactive program:

```
While True Echo;
```

This program replaces the usual first program of the traditional maths worldview approach:

```
Writeln('Hello World');
```

The Echo activity is an "atomic unit" of computation that continually reads its input and reproduces that signal on its output. It goes on forever until the program is killed. It interacts via a keyboard and screen with a user.

Looking inside this program, we may choose to divide it into two separate entities, one responsible for user input (the "source") and the other for output (the "sink"). For example Echo could be decomposed into read and write, the "source" entity is the reader and the "sink" entity is the writer.

In Pascal this would look as follows where `Readln` reads a line of input typed at the keyboard and `Writeln` writes a line of output to the screen:

```
While True
  Do
    Begin
      Readln(Input, InputString); {Input refers to the standard
                                   input stream or pipe}
      Writeln(Output, InputString);{Output refers to the standard
                                    output stream or pipe}
    End;
```

In pseudocode,

ECHOER

1. Read *something* from the user's input
2. Write that *something* to the user's screen

Which can be decomposed into SOURCE and SINK as follows:

SOURCE

1. Read *something* from the user's input and hand it to the SINK

SINK

1. On receipt of *something* from SOURCE, write it to the user's screen

These entities form a simple community, communicating with one another, but also with the user. Like other entities, humans are members of the community who interact with program components.

This shows the ways in which a single apparent entity (ECHOER) may in fact be constructed out of several entities cooperating. Each of these entities interacts with the other according to some predetermined protocol ("hand it to SINK"/"On receipt...") There are many ways to implement this interaction. It could be supplier-driven or it could be recipient-driven, for example.

If we create an entity that subscribes to both the source (producer) and sink (consumer) side of the protocol, we can insert a transformer entity into the community as shown in Figure ?.

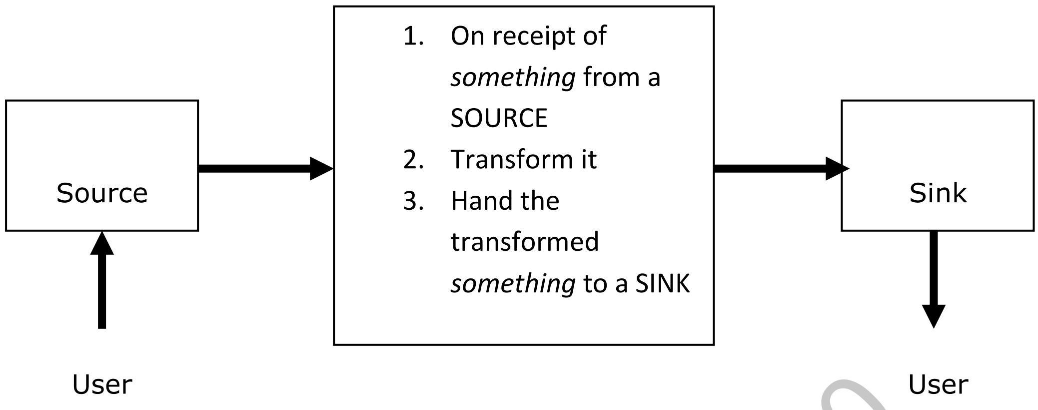


Figure ? Source-Transformer-Sink

Let’s suppose that the transformer is written to convert a temperature reading in Celsius into a temperature reading in Fahrenheit. If we place the transformer entity on a networked machine and communicate with it from another machine on the same network then we need an arrangement as shown in Figure ?

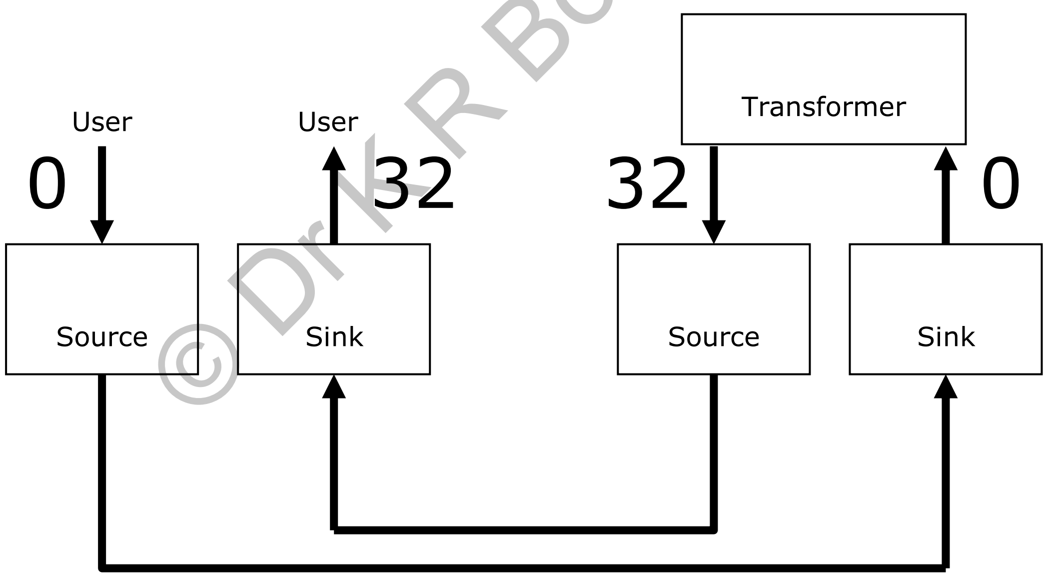


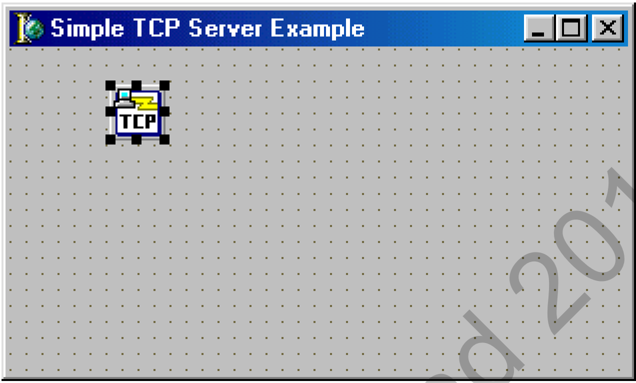
Figure ? Temperature Transformer

Projects

1. Build a client-server program to convert temperatures
2. Build a client-server chat program
3. Build a simple networked video game

Client-server program to convert temperatures (Key Stage 3)

1. Create a new Delphi project
2. Add to the main form a TIdTCPServer, located on the Indy servers' page of the Component palette. The main form is shown below



3. Set the IdTCPServer component's DefaultPort property to 5000 and its active property to True.
4. Double click this component's OnExecute event handler in the Object Inspector. And add the following code:

```
Procedure TForm1.IdTCPServer1Execute(AThread: TIdPeerThread);
  Var
    CelsiusTempString : String;
    CelsiusTempInteger : Integer;
  Begin
    With AThread.Connection
      Do
        Begin
          Try
            Writeln('Type a Temperature in degrees celsius and Press
                                                              Enter');
            CelsiusTempString := Readln;
            Try
              CelsiusTempInteger := StrToInt(CelsiusTempString);
              Writeln(CelsiusTempString + ' in Fahrenheit is ' +
                         IntToStr(Round(32 + (9* CelsiusTempInteger)/5)));
```

```
          Except
            Writeln(CelsiusTempString + ' is not an Integer');
          End;
        Finally
          Disconnect;
        End;

      End;
End;
```

5. Disable the integrated debugger, temporarily, by selecting Tools | Debugger Options, from Delphi's main menu if you are running the server from inside the IDE.
6. Now save your project and run it.
7. Open a MS-DOS prompt/console window.
8. Begin a Telnet session and connect to your simple server by entering the following command at the command prompt:

Telnet 127.0.0.1 5000

9. Now go to another machine on the network, logon and begin a Telnet session as before but this time type:

Telnet IP Address of the machine on which the Simple TCP Server is running 5000

10. Now add the following code marked in bold to the event handler:

```
AssignFile(Pipe,
              'N:\MyWork\LessonsOnNetworking\SimpleTCPServer\Log.Txt');
Append(Pipe);
Writeln(Pipe, AThread.Connection.Socket.Binding.IP, ' ',
                        AThread.Connection.Socket.Binding.Port);
{Check your version of Delphi - the code for AThread may need to change}
CloseFile(Pipe);
With AThread.Connection
  Do
    Begin
      Try
        Writeln('Type a Temperature in degrees celsius and Press Enter');
        CelsiusTempString := Readln;
        Try
          CelsiusTempInteger := StrToInt(CelsiusTempString);
          Writeln(CelsiusTempString + ' in Fahrenheit is ' +
                 IntToStr(Round(32 + (9* CelsiusTempInteger)/5)));
        Except
          Writeln(CelsiusTempString + ' is not an Integer');
```

```
    End;
  Finally
    Disconnect;
  End;
```
11. Add a Pipe declaration as follows:

```
Var
        Form1: TForm1;
        Pipe : TextFile;
```

12. Save your project.
13. Open WordPad or NotePad and create an empty text file with the file name 'Log.Txt' in the same folder as this Delphi project.
14. Run your project and then Telnet as before. When you have finished open the 'Log.Txt' file in WordPad and note that a log has been created of the IP address and port number of all hosts which have Telnetted into your server.

Client-Server Chat program (Key Stage 4)

Create a new Delphi forms-based project.

Add UDP.Pas and UDP.FRM to your Delphi project. UDP.FRM contains two Indy components IdUDPClient1 and IdUDPServer1 as shown in Figure ?

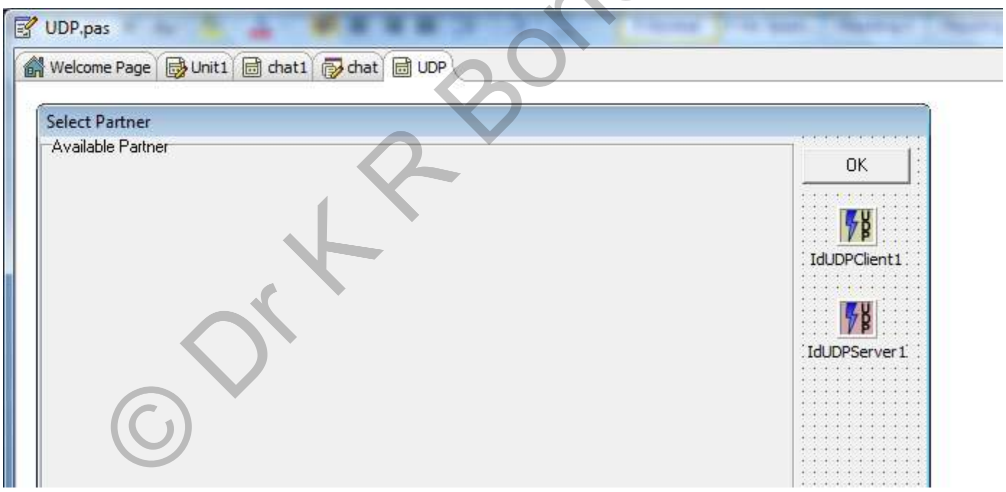
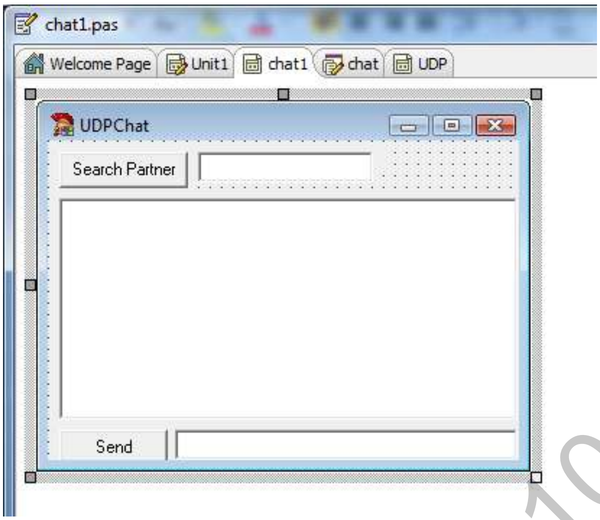


Figure ? UDP.Frm

Add the following to the form as shown in Figure ?:

1. Memo box
2. Two buttons
3. Two edit boxes
4. One label

Place the caption "Search Partner" on the first button and the caption "Send" on the second button.



Double click the Search Partner button (Button1) and add the code to the body of the event handler TForm1.Button1Click shown below. Double click the Send button (Button2) and add the code to the body of the event handler TForm1.Button2Click shown below.

Create an event handler for the OnActivate event for Form1 – use Object Inspector Events tab

Add to the class definition for TForm1 as shown:

```
Activated: Boolean;
Procedure SearchEvent(ResultIP, ResultName: String);
Procedure UDPRead(Sender: TObject; AData: TStream;
                  ABinding: TIdSocketHandle);
Procedure UDPException(Sender: TObject);
```


Add Uses UDP;

```
Interface

Uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, IdSocketHandle, StdCtrls;
Type
  TForm1 = Class(TForm)
           Public
             Button1: TButton;
             Memo1: TMemo;
             Edit1: TEdit;
```

```
        Button2: TButton;
        Edit2: TEdit;
        Label1: TLabel;
        Procedure Button1Click(Sender: TObject);
        Procedure FormActivate(Sender: TObject);
        Procedure Button2Click(Sender: TObject);
    Private
      Activated: Boolean;
      Procedure SearchEvent(ResultIP, ResultName: String);
      Procedure UDPRead(Sender: TObject; AData: TStream;
                        ABinding: TIdSocketHandle);
      Procedure UDPException(Sender: TObject);
    End;


Implementation


Uses UDP;


{$R *.dfm}


Procedure TForm1.Button1Click(Sender: TObject);{Search Pattern
button}
  Begin
    UDPSearchForm.SearchEvent := SearchEvent;
    UDPSearchForm.Left := Left;
    UDPSearchForm.Top := Top
    UDPSearchFormAktIP := Edit1.Text;
    UDPSearchForm.SearchPartner;
     End;
Procedure TForm1.Button2Click(Sender: TObject); {Send button}
  Var
    x: Array[0..100] of Byte;
    i: Integer;
  Begin
    UDPSearchForm.Host := Edit1.Text;
    UDPSearchForm.Active := True;
    x[0] := $10; // Text
    x[1] := 0;   // Type 0
```

```
    For i := 1 To Length(Edit2.Text)
      Do x[i+3] := Byte(Edit2.Text[i]);
    UDPSearchForm.DoSend(x, 4+Length(Edit2.Text), Length(x));
  End;
```

Add the code

```
Procedure TForm1.SearchEvent(ResultIP, ResultName: String);
  Begin
    Edit1.Text := ResultIP;
    Label1.Caption := ResultName;
  End;

Procedure TForm1.UDPRead(Sender : TObject; AData : TStream;
                         ABinding : TIdSocketHandle);
  Var
    Buffer: Array [0..2047] of Byte;
    Count: Integer;
    PeerIP: String;
    PeerPort: Integer;
    s, ss: String;
    i: Integer;
  Begin
    PeerIP := ABinding.PeerIP; {Get IP address of remote machine}
    PeerPort:= ABinding.PeerPort;
    Count := AData.Size; {Get how many bytes received}
    If Count > Length(Buffer)
      Then Exit;
    AData.Read(Buffer, Count); {read received bytes into buffer}
    If (Buffer[0] <> $00) And  (Buffer[0] <> $01)
      Then Edit1.Text:= PeerIP;{OK got text from remote machine}
   Case Buffer[0] Of
     $00: If Count = 4 {$00 is a search request}
            Then
              If Buffer[1] = 0
                Then
                  Begin
                    Buffer[0] := $01;
                    UDPSearchForm.Host := PeerIP;
```

```
                    UDPSearchForm.DoSend(Buffer, 4,
                                           Length(Buffer));
                   {Reply with IP address of this machine}
                    Memo1.Lines.Add('Inquiry [' +
                             UDPSearchForm.WSGetHostByAddr(PeerIP) +
                          '(' + PeerIP + ')' +          ' Port: ' +
                          IntToStr(PeerPort) + ']');
                  End;

      $01: If Count = 4 {$01 is a search reply}
             Then
               If Buffer[1] = 0
                 Then
                  Begin
                    ss := UDPSearchForm.WSGetHostByAddr(PeerIP);
                    s := '[' + ss + '(' + PeerIP + ')' +
                        ' Client Port: ' + IntToStr(PeerPort) + ']';
                    Memo1.Lines.Add('Inquiry Reply ' + s);
                    If PeerIp = UDPSearchForm.LocalAddress
                      Then ss := '<myself>' + ss;
                    UDPSearchForm.Add(PeerIP, ss);
                  End;

      $10: If Buffer[1] = 0 {$10 is text}
             Then
               Begin
                 s := '';
                 For i := 4 To Count-1
                   Do s := s + Char(Buffer[i]);{Extract the text}
                 Memo1.Lines.Add(PeerIP+'->' + s);
               End;
    End;
End;

Procedure TForm1.UDPException(Sender: TObject);
  Begin
  //nothing
  End;
Procedure TForm1.FormActivate(Sender: TObject);
```

```
Var
  s, s2: String;
Begin
  If Activated Then Exit;
  Memo1.Clear;
  Activated := True;
  UDPSearchForm.OnUDPRead := UDPRead;
  UDPSearchForm.OnException := UDPException;
  UDPSearchForm.Active := True;
  s := UDPSearchForm.LocalAddress;
  s2 := UDPSearchForm.WSGetHostByAddr(s);
  Memo1.Lines.Add('I''m (' + s + ') ' + s2);
End;
```

Compile and run the project . The executable will find the machine running it. You need to know the IP address of the machine you wish to chat with. This machine must be running a copy of the executable. A result similar to that shown in Figure ? is obtained when two networked machines communicate.

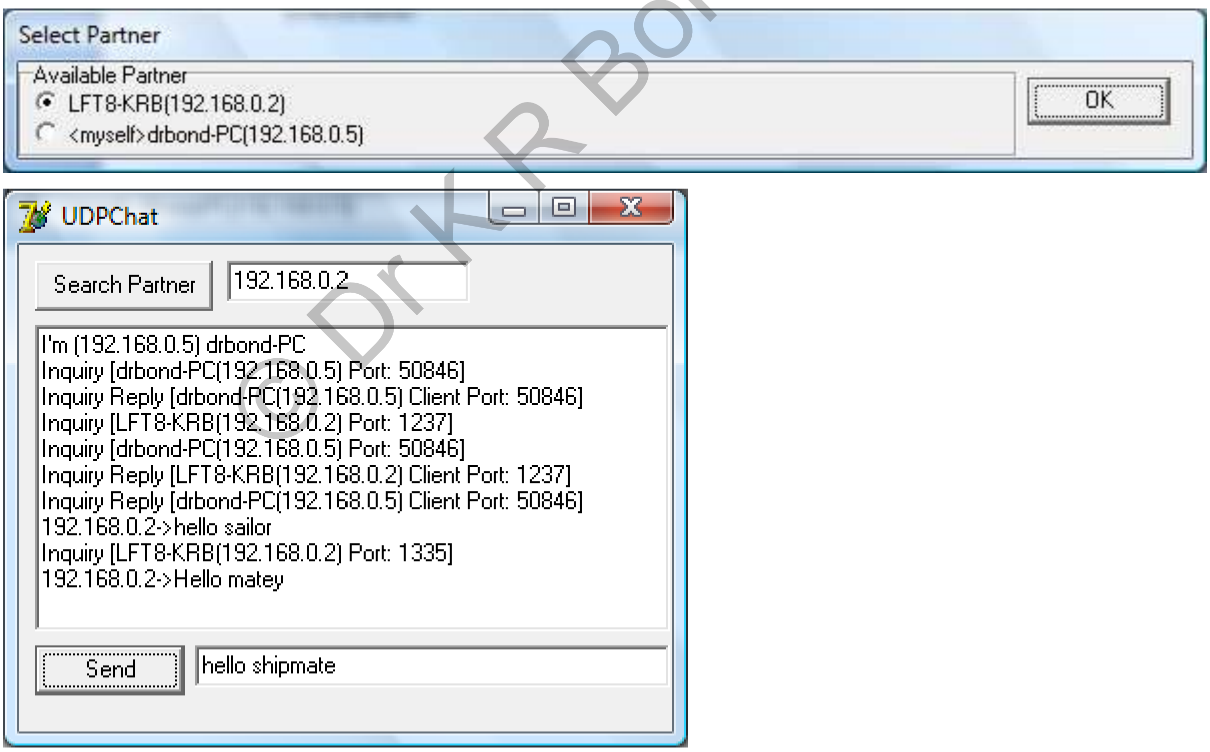


Figure ? Client-server Chat program

Secret Messages and Crypotography

Encryption Spreadsheet Exercise

Learning Objectives	**Explanation**	**Page reference**
Understanding encryption	Plain text to cipher text	2
Modular arithmetic	Modulo arithmetic using numerals 0 to 25	4
ASCII character codes	A - 65, B - 66	2
Using a formula	=	4
Using functions	SUM(), MOD(), IF(), CODE(), CHAR()	4, 6, 9
Finding modular inverses	Modulo arithmetic using numerals 0 and 1	5
IF()	If Then Else	9

What is Cryptography?

Cryptography is the science of sending secret messages. People have been sending secret messages for thousands of years. People shopping on the Internet use them to keep their credit card numbers secret. The military send secret messages so their plans will not be revealed to the enemy.

In cryptography, the word **cipher** is used to mean a particular method of producing a secret form of a message:

The method changes one letter of the message into another letter or symbol.

In the **Caesar cipher** the alphabet is shifted a certain number of places and each letter is replaced by the corresponding shifted letter. For example shifting the alphabet two places to the left gives the Caesar cipher shown in **Table 1**.

Plaintext	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher text	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B

Table 1 Plaintext encrypted to produce cipher text

Plaintext and Cipher Text

Encryption means turning a plaintext message such as **RED GUN** into an unintelligible (gobbledygook) encrypted form called cipher text which in this case is **ZMJ SIN** **(in this example the cipher used is not the Caesar cipher)**

Decryption

Decryption means turning the cipher text message **ZMJ SIN** into a form called the plaintext message which is intelligible, in this case RED GUN

Can you decrypt the encrypted message below?

YGNN FQPG

During World War II, the German armed forces communicated secretly using a system of messages encrypted with Enigma machines. The Allies were greatly aided in their attempts to decrypt these secret messages by the capture of an Enigma machine. They were able to reverse engineer the captured Enigma machine and to confirm how the plaintext messages were converted into encrypted messages. This greatly assisted the construction of computer programs[15] for the Colossus computer to decrypt future intercepted encrypted messages.

PC Activity

An Enigma simulator can be found at http://cryptocellar.org/simula/. Use this to encrypt and decrypt messages and to see the internal settings of the machine for your messages. Do a web search for background information on the Enigma machine and Bletchley Park, the home of the British code breaking attempts in World War II.

15The Colossus machine was an electronic computing device used by British code breakers to read encrypted German messages during World War II. The instructions for these programs were placed in the computing machine by actually rewiring the machine's control store.

Figure 1 Enigma machine simulator

Remember

Information hiding means hiding design details behind a standard interface.

Figure 1 shows an Enigma machine simulator. Its internal design is a complex system of rotors and wiring hidden behind a nearly standard QUERTY keyboard. This keyboard is the **interface** between the user and the internal operation of the machine. The operator does not need to know how the operation of the machine is achieved only how to control it.

An interface: is a boundary between the implementation of a system and the world that uses it. It provides an abstraction of the entity behind the boundary thus separating the methods of

Interfaces are important for another reason. By hiding the complexities of the machine behind a well-designed interface users are able to learn how to use a machine with the minimum of training. The design of the motorcar is a classic example. Motorcars present a standard interface (pedals, gear shift, instrument gauges, light switches) on which people can be trained and licensed without needing to know anything about the internal design and operation of the internal combustion engine, transmission systems and gearboxes. What is more a licensed driver does not need to learn a completely different way of driving every time they drive a new model (accepting that there are slight differences between manual and automatic transmissions).

Steganography

How can you count to 1023 using just your eight fingers and two thumbs?

Solution is on next page.

An image is represented by a bitmap inside a computer - see **Figure 2**. A bit map is just a sequence of the numerals **0** and **1** stored in an area of a computer's RAM. The binary system of counting uses just the two numerals **0** and **1**. Numerals **0** and **1** are labelled **binary digits** or **bits** (**B**inary dig**ITS**), hence the term bitmap.

1	0	0	0	1	0	1	0	0
1	0	1	0	0	1	1	1	1
1	0	1	0	0	1	1	1	0
1	1	1	0	0	1	0	1	1
1	1	1	0	0	1	1	1	1
0	0	1	1	0	1	1	0	1
1	1	1	0	1	1	1	1	0

Figure 2 An image and its bit map

Plaintext messages are converted into sequences of binary digits or bits. It is very easy to hide messages in images, just replace the bits in the image's bitmap by the message bits. There is a secret message hidden in the image in **Figure 2**. This is called **steganography**.

Solution

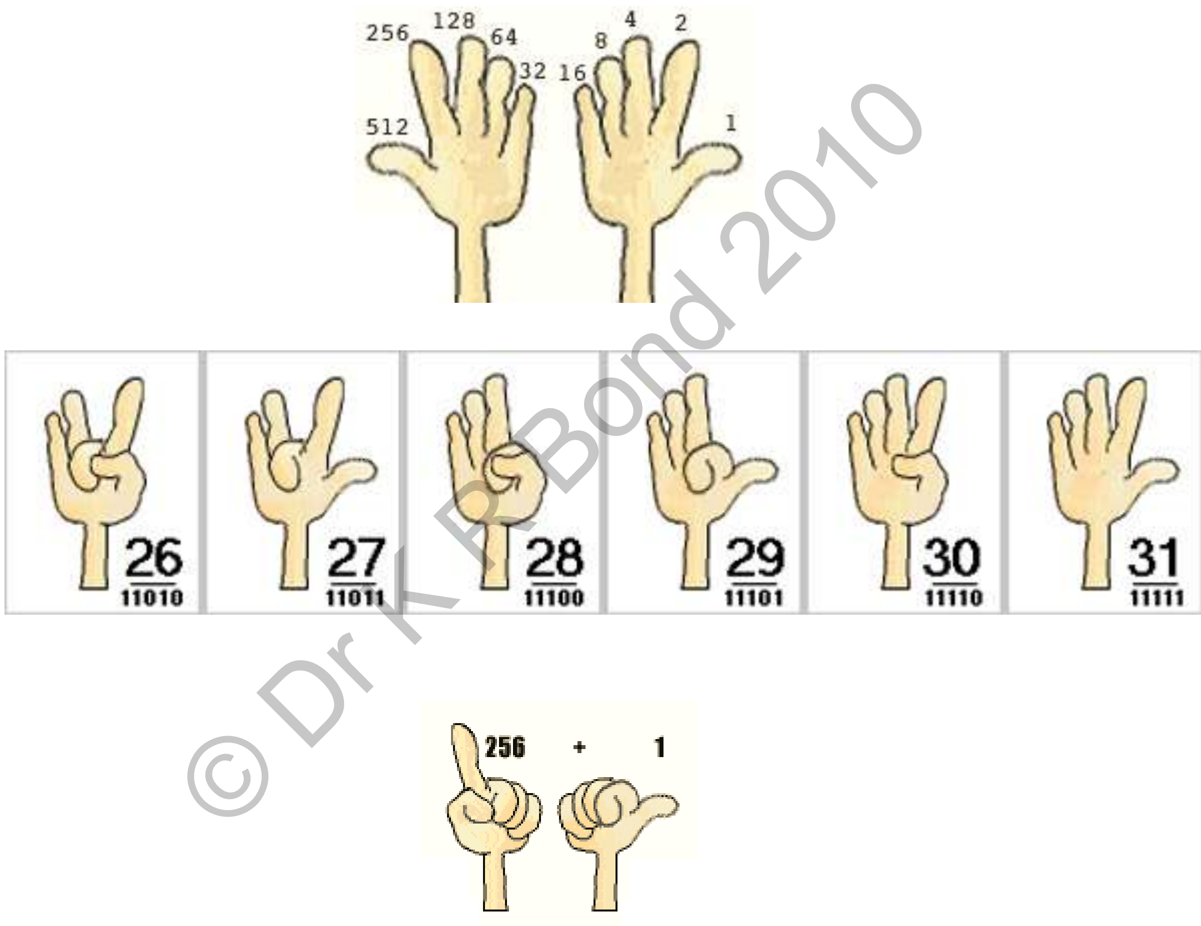


Figure 3 Binary Finger Counting

Changing letters to numbers

STEP 1: Open a new WordPad document. Press CAPS LOCK key. Test that when you type ABCDE, letters appear in upper case in the WordPad document. Now, whilst holding down the ALT key, type 65 on the number pad. The letter A should appear in your WordPad document. What number should you type on the number pad to get the letter B?

Why does this work?

Answer: The keyboard generates a **number** when the **letter key A** is pressed and a **different number** when the **letter B** is pressed. Therefore, it should be possible to make an **A** and a **B** appear in the WordPad document by typing the numbers corresponding to **letter A** and **letter B** on the number pad as long as we tell the keyboard that we are using an alternative approach to entering letters, i.e. the **ALT key** is pressed at the same time.

STEP 2: We can use a spreadsheet to find out which numbers are assigned to which letters. Enter the letter **A** into cell **A5**. Enter the formula

=CODE(A5)

into cell **B5**. This formula changes the letter into a number. **Figure 4** shows the result (ignore contents of cell C3 and E3).

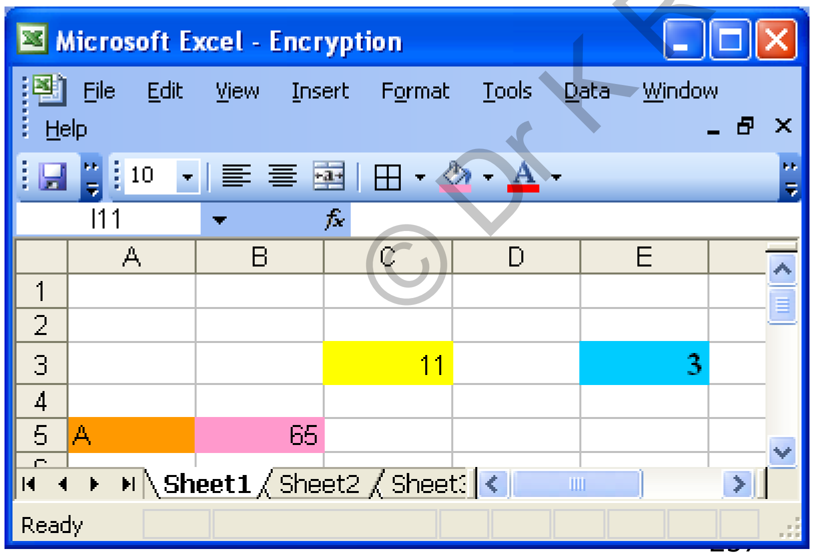
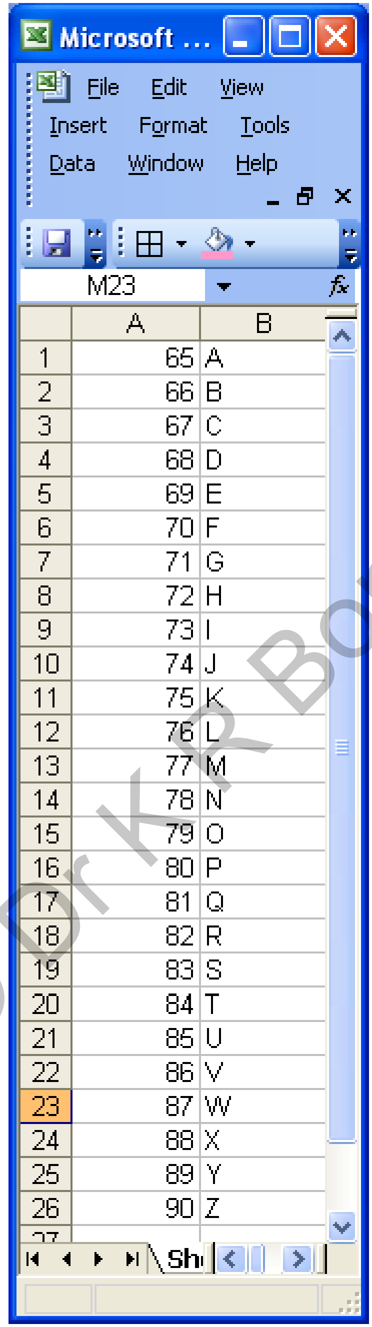


Figure 4 Changing a letter into a number

STEP 3: Replace the letter **A** by the letter **B** then **C** and so on. Note that letter **B** changes to the number **66**.



	A	B
1	65	A
2	66	B
3	67	C
4	68	D
5	69	E
6	70	F
7	71	G
8	72	H
9	73	I
10	74	J
11	75	K
12	76	L
13	77	M
14	78	N
15	79	O
16	80	P
17	81	Q
18	82	R
19	83	S
20	84	T
21	85	U
22	86	V
23	87	W
24	88	X
25	89	Y
26	90	Z

Figure 5 shows a spreadsheet containing a **change letter to number formula** for the **twenty six letters** of the alphabet.

STEP 4: Change your spreadsheet so that your spreadsheet shows a similar result.

Figure 5 Changing letters of the alphabet into numbers

Simple Encryption using the number 2 as Key

STEP 5: Enter the formula **= A1 + 2** into cell **C1**. Copy and paste this formula into cells **C2** to **C26**

STEP 6: Enter the formula **=CHAR(C1)** into cell **D1**. Copy and paste this formula into cells **D2** to **D26**.

This formula uses a function named **CHAR**. This turns a number into its equivalent letter or keyboard key symbol, e.g. **65** is turned into the letter **A**, **91** into the symbol **[**

Your spreadsheet now shows that the letter **A** has been encrypted as the letter **C**, the letter **B** as **D** and so on. This is what happens when we use the number 2 as the key.

STEP 7: Try a different number for the key (don't make your key bigger than 36).

Clock/Modular arithmetic

The clock shown in **Figure 3** has just two numerals **0** and **1**. After one hour has elapsed the hour hand has moved half way round the clock face and therefore points at numeral **1**. After another hour has elapsed the hand has rotated half way round the clock face again, pointing now at numeral **0**. After another hour has elapsed the hour hand now points at numeral **1**.

The answer to the question where does the hour hand point after 5 hours is numeral **1**.

To reach numeral **1** after 5 hours the hand must rotate from **0** to **1**, **1** to **0**, **0** to **1**, **1** to **0** and finally **0** to **1**.

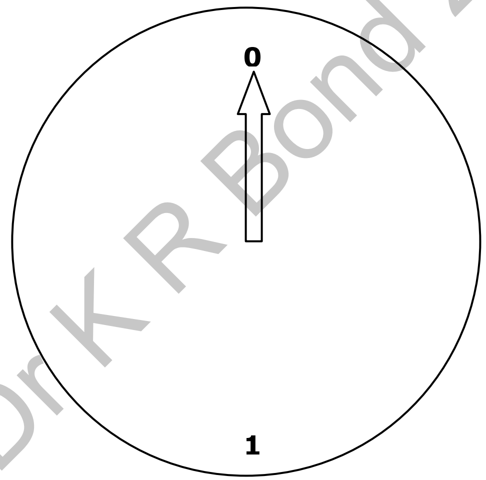


Figure 3 Clock with just two numerals 0 and 1

Spreadsheet function to calculate clock result with just two numerals

The **MOD function** performs clock arithmetic. Tell it how many numerals are used on the clock face, e.g. 2 and how many hours have passed and it will tell the time using just these numerals, 0 and 1.

For example, **= MOD(5, 2)** produces the answer **1** because the number of hours that have passed is given as **5**.

What is the answer when the number of hours passed is

(a) 7
(b) 8?

When is 2 + 2 not equal to 4?

2 plus 2 is always **4**, **4** plus **4** is always **8** and **8** plus **8** is always **16**.

Can you think of an example to prove that **2 plus 2** is not always **4**?

(Hint: If it is **10pm** now and you want to sleep for **8** hours, what time should you set your alarm to ring? **6 am**. Therefore, **10 + 8 = 6** not **18**!)

Now if our clock has only four numerals on its face then **2 + 2 = 0**. If the clock hand rotates through **4 hours** then it arrives back at **0** as shown in **Figure 4**. It is a **four-hour clock**.

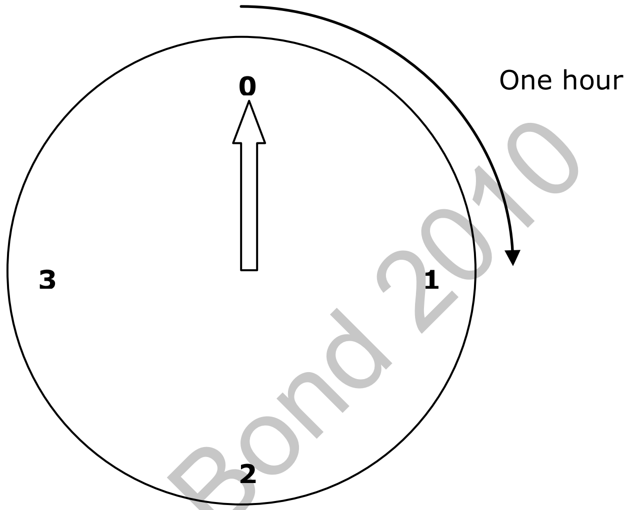


Figure 4 clock with just four numerals

Set up a spreadsheet and enter the formula

= MOD(C3, 4) into cell E3

Enter a value, e.g. 4 into cell C3. Observe the result that is shown in cell E3. **Figure 5** shows a spreadsheet that has been set up as described.

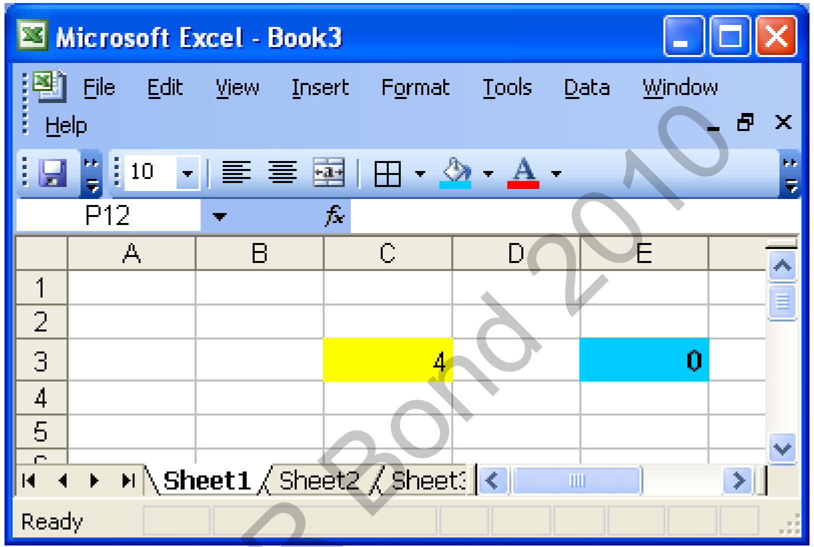


Figure 5 Modular arithmetic example using 4 numerals 0, 1, 2, 3

Questions

Use clock/modular arithmetic to solve these problems for four numerals 0, 1, 2, 3

(a) 5 + 10
(b) 9 + 8 + 7
(c) 3 - 7 = (Hint move backward around the clock)

Use the spreadsheet set up as for **Figure 5** to check your solutions.

Clock size

We used a clock of **size 4** in **Figure 4**, i.e. a clock with four numerals. The word **modulus** or **mod** is used to show which clock size was used.

For example, the expression

4 + 3 mod 4 means use a 4-hour clock

and the expression

27 mod 9 means use a 9-hour clock

We write **4 +3 = 3 (mod 4)** to make it clear that we used clock/modular arithmetic.

Question

List all the numbers between **0** and **19** that have the same position on the **4-hour clock**.

Use the spreadsheet set up as for **Figure 5** to check your solutions.

Congruence

There is a special term in modular arithmetic to describe numbers that have the same position on a clock.

Two numbers are **equivalent mod n** if they differ by a **multiple of n**, i.e. if they have the **same position on a clock of size n**.

Table 1 shows numbers which are **equivalent mod n** for a **clock of size 4**.

Clock face numeral	Equivalent mod 4	Equivalent mod 4	Equivalent mod 4	Equivalent mod 4
0	0	4	8	16
1	1	5	9	17
2	2	6	10	18
3	3	7	11	19

Table 1 numbers which are equivalent mod n for a clock of size 4.

The symbol ≡ means **is equivalent to**. Using this notation,

$$11 \equiv 3 \pmod 4$$

The notation uses **mod 4** in **parentheses** to tell us what clock the numbers are on.

Another term for **equivalent mod n** is **congruent mod n**.

You can find numbers which equivalent or congruent to another number by adding multiples of the modulus. For example, **13**, **25**, **37** and **49** are all **equivalent mod 12 to 1** since they are all **1 plus a multiple of 12**.

$$1 + 1 \times 12 = 13$$

$$1 + 2 \times 12 = 25$$

$$1 + 3 \times 12 = 37$$

$$1 + 4 \times 12 = 49$$

Questions

List three numbers equivalent to each number

(a) 6 mod 12
(b) 9 mod 12
(c) 11 mod 12

Change your spreadsheet (**Figure 5**) and check your answers.

Reducing mod n

When we **work mod n**, we often only use the numbers from **0 to n -1**. If another number comes up, we **reduce mod n** which means we replace it with the number between **0 and n - 1** that is **equivalent mod n** to it. This is the remainder when we divide by **n**.

For example, **37 is equivalent mod 12** to the numbers **1**, **13**, **25** and so on. Of these, the number in the range from **0** to **11** is **1**, so **reducing 37 mod 12** gives **1**.

Example

Let's **reduce 26 mod 12**

Since we are **working mod 12**, we need to find the number from **0 to 11** that is equivalent to **26 mod 12**. One way to do this is to **subtract 12 repeatedly** until we get a number between **0** and **11**.

$$\begin{array}{r} \mathbf{26} \\ -\ \mathbf{12} \\ \hline \mathbf{14} \\ -\ \mathbf{12} \\ \hline \mathbf{2} \end{array}$$

We stop when we get to a number less than **12**, in this case **2**.

Questions

Reduce each number.

(a) 8 mod 6 (b) 13 mod 6 (c) 6 mod 6 (d) 5 mod 6 (e) 177 mod 26 (f) 107 mod 26 (g) 65 mod 26

Change your spreadsheet (**Figure 5**) and check your answers.

Inverses

In regular arithmetic, the way to undo multiplication by **3** is to divide by **3**.

We can show this with arrows,

$$\mathbf{5 \xrightarrow{x3} 15 \xrightarrow{\div 3} 5}$$

or as an equation,

$$\mathbf{(5 \text{ x } 3) \div 3 = 5}$$

Another way is to multiply by ⅓ since multiplying by ⅓ is the same as dividing by **3**.

$$\mathbf{5 \xrightarrow{x\ 3} 15 \xrightarrow{x\ 1/3} 5}$$

The arrows show that we start and end with the same number.

We can also show this as an equation:

$$(5 \times 3) \times \tfrac{1}{3} = 5$$

Multiplying first by **3** and then by **⅓** gives us back what we started with!

We can write the above equation as follows:

$$5 \times (3 \times \tfrac{1}{3}) = 5$$

i.e. **5 x (3 x ⅓)** is the same as **5 x 1** because **(3 x ⅓) = 1**

The multiplicative inverse of **3** is the number n that solves

$$3 \times n = 1$$

Therefore, the multiplicative inverse of **3** is **⅓.**

What is the multiplicative inverse of the following numbers?

(a) 5
(b) 15

In regular arithmetic the multiplicative inverse of a number is its **reciprocal**.

Modular inverses

The **mod 26 inverse of 3** is the number from **0** to **25** that solves

$$\mathbf{3 \times n = 1 \ (mod\ 26)}$$

Table 2 shows the result of multiplying **3** by **n** and that the modular inverse of **3** is **9**, i.e. **3 x 9 mod 26 = 1**

3 x n = 1 (mod 26)
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 1 (mod 26)

Table 2 Modular inverses

Getting back to where you started

Testing **9** to see if it works like an inverse:

$$\mathbf{4 \xrightarrow{x3} 12 \xrightarrow{x9} 108 = 4 \ (mod\ 26)}$$

Try the following:

1. $\mathbf{6 \xrightarrow{x3} 18 \xrightarrow{x9} 162 = \ ? \quad (mod\ 26)}$

2. $\mathbf{2 \xrightarrow{x3} \ ? \xrightarrow{x9} \ ? \quad = \ ? \quad (mod\ 26)}$

3. $\mathbf{10 \xrightarrow{x3} \ ? \xrightarrow{x9} \ ? = \ ? \quad (mod\ 26)}$

Encryption of Plaintext Using Modular Arithmetic

We are going to use the encryption key **3** to encrypt the letters of the plaintext message

MEET UNDER THE CLOCK TOWER TONIGHT

Figure 6 Plaintext message

STEP 1: For each letter in the message in **Figure 6** write down the corresponding number of the letter using **0** for letter **A**, **1** for letter **B** and so on (ignore spaces, i.e. don't translate the spaces). Place your numbers in the first row of **Table 3 -** some numbers have been entered already. **Table 4** on the next page may help you do this correctly.

	4	4		█	20			4		█			4	█						█				4		█							
M	E	E	T		U	N	D	E	R		T	H	E		C	L	O	C	K		T	O	W	E	R		T	O	N	I	G	H	T
	12	12		█	8			12		█			12	█						█				12		█							
	M	M		█	I			M		█			M	█						█				M		█							

Table 3 Blank table for your plaintext, plaintext numbers, your encrypted numbers and encrypted text

Letters	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
Numbers	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**

Table 4 Letter to number table

STEP 2: Now with the help of **Table 5**, write down for each number

number x 3 (mod 26)

in the blank spaces in the third row of **Table 3**. Some numbers have been done for you already.

Numbers	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
x 3 (mod 26)	0	3	6	9	12	15	18	21	24	1	4	7	10	13	16	19	22	25	2	5	8	11	14	17	20	23

Table 5 Number x 3 (mod 26)

STEP 3: Now using **Table 4** write down in **row 4** of **Table 3** the letter corresponding to each number in **row 3** of this table.

Decryption of Encrypted Message Using Modular Arithmetic Inverses

We are going to use the decryption key **9** to decrypt the letters of the encrypted message.

STEP 1: Copy the encrypted message letters into **Table 6**. Convert these letters into numbers using **Table 7**.

	M	M		■	I			M		■			M	■						■				M		■							
	12	12			8			12					12											12									
	4	4			20			4					4											4									
	E	E			U			E					E											E									

Table 6 Decryption table

Letters	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
Numbers	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**

Table 7 Letters to numbers

STEP 2: Now with the help of **Table 8**, write down for each number

number x 9 (mod 26)

in the blank spaces in the third row of **Table 6**. Some numbers have been done for you already.

Numbers	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
x 9 (mod 26)	0	9	18	1	10	19	2	11	20	3	12	21	4	13	22	5	14	23	6	15	24	7	16	25	8	17

Table 8 Numbers x 9 (mod 26)

STEP 3: Now with the help of **Table 7** write down in **row 4** of **Table 6** the letter corresponding to each number in **row 3** of this table. Some numbers have been converted for you.

Communications

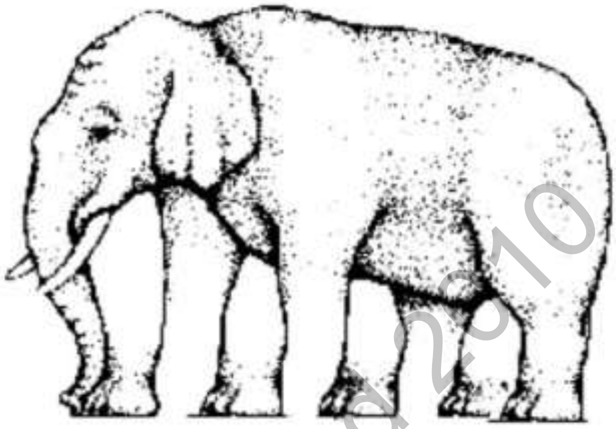
Key Stage 2
Covers principles 1.1.1 – 1.1.7, 3.1.1,

Lesson Plan

1. Introduction
2. Communicating
    a. Images
        i. Flags
        ii. Optical illusions
    b. Morse code
    c. Christmas lights
    d. Binary
    e. Keyboard codes for text
    f. Encryption
    g. Images and pixels
3. Error correction
4. Wii controller

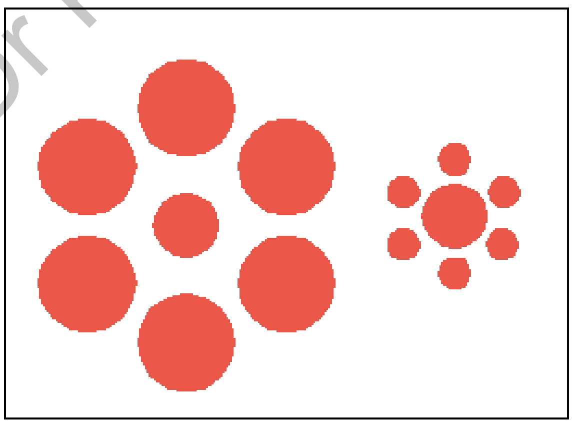
1. Things may not be what they seem to be. Let's start with a little quiz:

   a. Who wrote Beethoven's fifth symphony?
   b.



How many legs does this elephant have?

   c. Which circle in the middle is bigger?



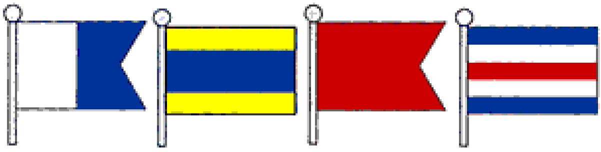
   d. When is 10 not 10?

2. Communicating

a. Images

i. Flags

The International Code of Signals was introduced in 1857. The original Code contained 17,000 signals using 18 signal flags.



A B C D

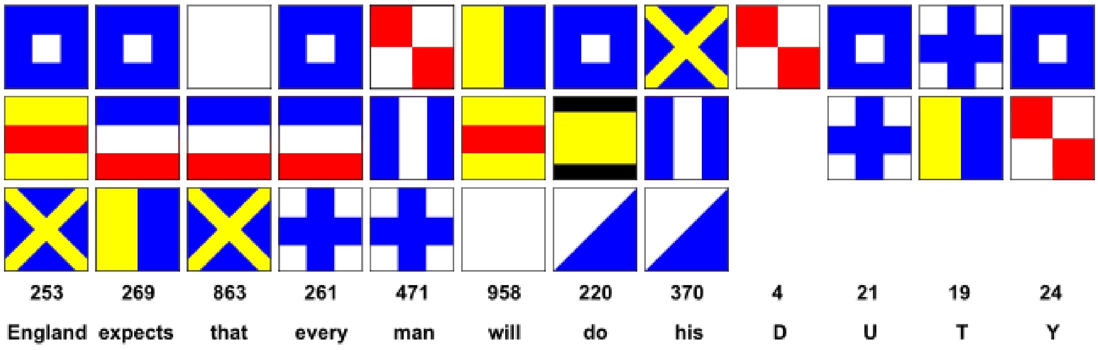
The Battle of Trafalgar

At 11:45 on October 21, 1805 one of the most famous naval signals in British history was sent. The message was composed by Admiral Lord Nelson.

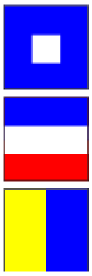
"England expects every man to do his duty."

This signal was relayed using the numeric code devised by Sir Home Popham. This code assigned the digits 0 to 9 to ten signal flags. These flags in combination represented code numbers which were assigned meaning by a code book, distributed to all Royal Navy ships and weighted with lead for disposal overboard in case of capture.

(Need permission to reproduce this image)



Write alongside each flag below the number that the following flag represents. The number will be between 0 and 9, inclusive.



Is there an error in Admiral Lord Nelson's flag message above?

ii. Optical illusions

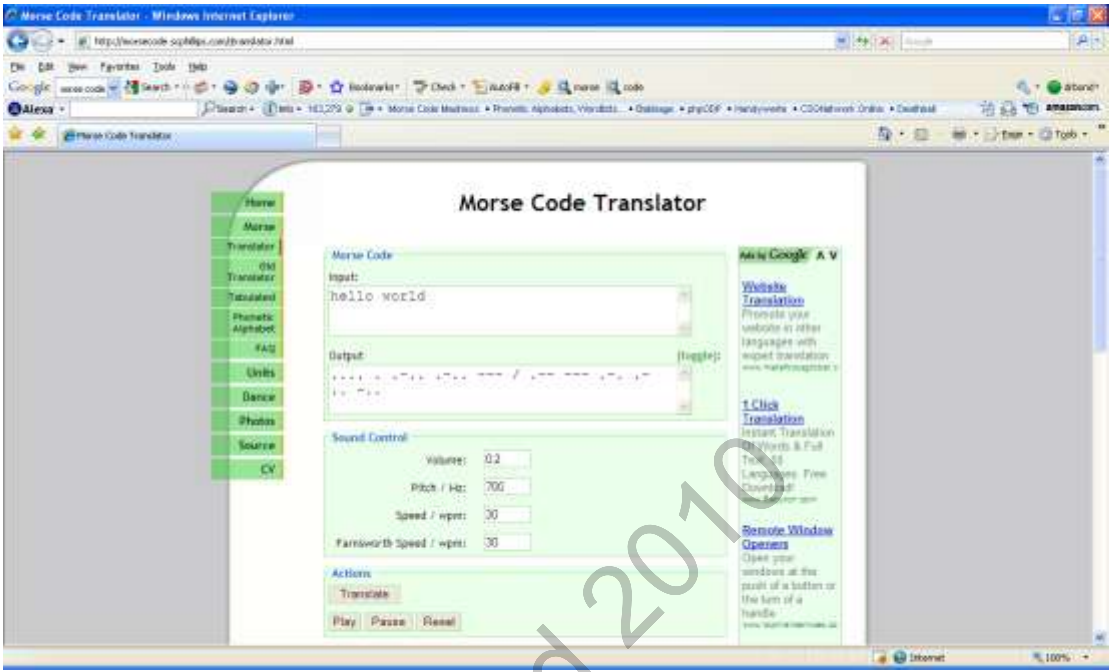
Communicating by flag has weaknesses:

- Doesn't work at night in the dark
- Doesn't work when visibility is poor because of fog
- Doesn't work over long distances
- The eye can play tricks:

You should see a man's face and also a word...
Hint: Try tilting your head to the right, the world begins with 'L'

b. Morse code

http://morsecode.scphillips.com/jtranslator.html



Morse Code Alphabet

The International Morse code characters are:

A .-	**N** -.	**0** -----
B -...	**O** ---	**1** .----
C -.-.	**P** .--.	**2** ..---
D -..	**Q** --.-	**3** ...--
E .	**R** .-.	**4** ....-
F ..-.	**S** ...	**5** .....
G --.	**T** -	**6** -....
H	**U** ..-	**7** --...
I ..	**V** ...-	**8** ---..
J .---	**W** .--	**9** ----.
K -.-	**X** -..-	**Fullstop**
L .-..	**Y** -.--	.-.-.-
M --	**Z** --..	**Comma**
		--..--

c. Christmas lights

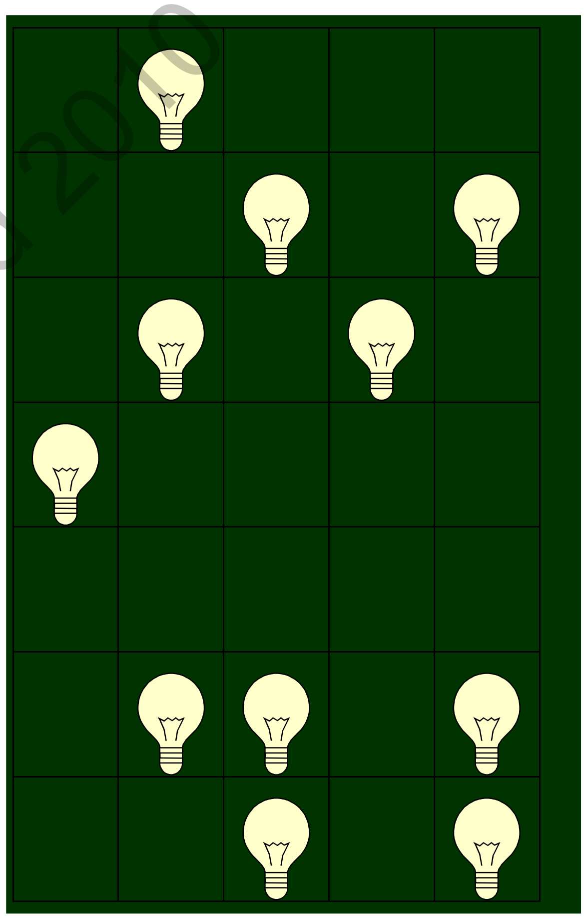
John found himself locked in a department store on Christmas Eve, 1998. How did he manage to be rescued?

Well, on the second floor he discovered boxes and boxes of Christmas tree lights. He noticed that there were people still working in the building across the road.

How could he signal to these people that he was trapped in the department store? The road outside was very busy with noisy traffic, too noisy to shout HELP!

This is what he devised using Christmas tree lights:

Letter	**Decimal Number**	**Binary**
	8	01000
	5	00101
	12	01010
	16	10000
	0	00000
	13	01101
	5	00101



Can you devise a code for the letters of the alphabet, A to Z and a space?

Can you decode the message?

Letter	Decimal Number	Letter	Decimal Number
SPACE	0	N	14
A	1	O	15
B	2	P	16
C	3	Q	17
D	4	R	18
E	5	S	19
F	6	T	20
G	7	U	21
H	8	V	22
I	9	W	23
J	10	X	24
K	11	Y	25
L	12	Z	26
M	13		

d. Binary

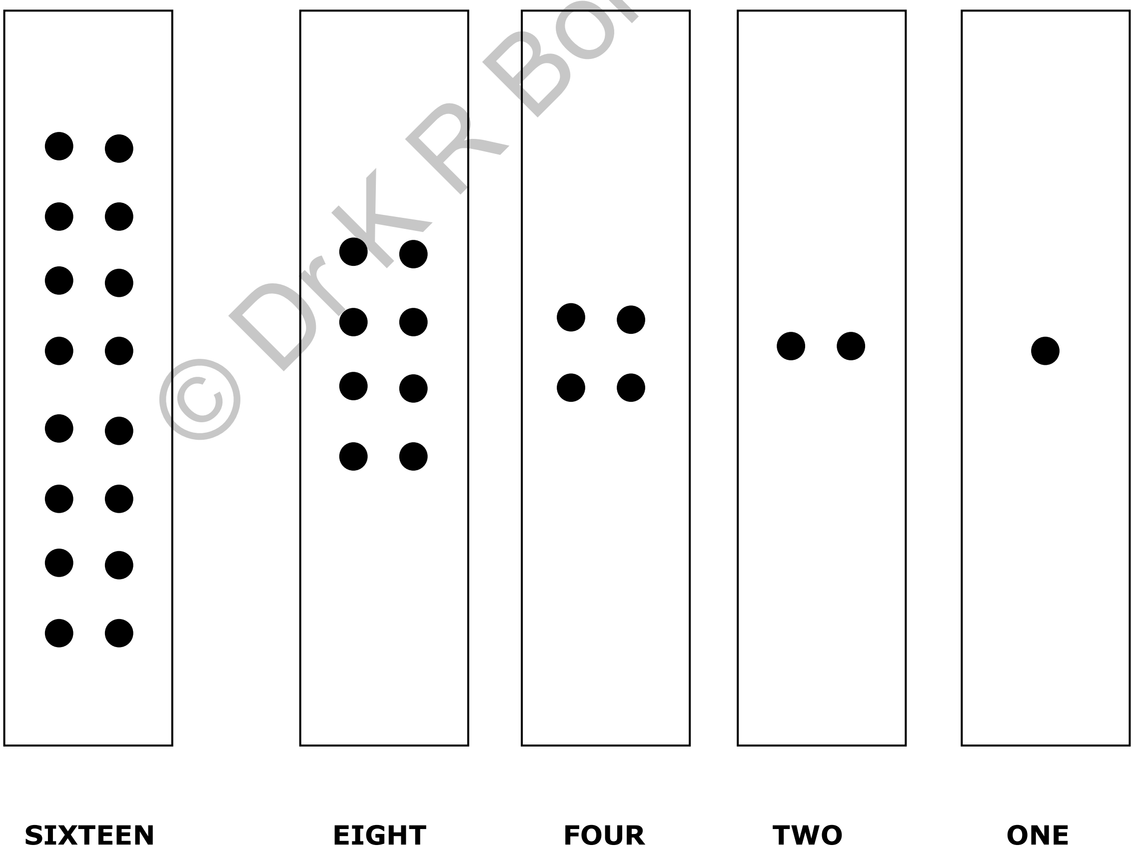


Figure 2 shows how to use the fingers and thumbs of your two hands to count in binary.

Figure 2

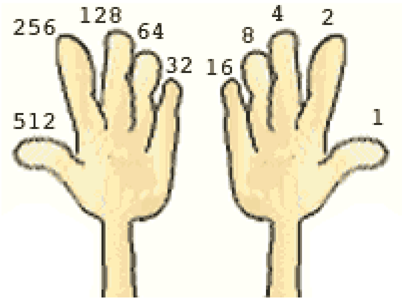
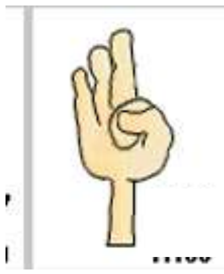


Figure 3



What does the hand in **Figure 3** represent expressed in binary?

A. 00011 **B.** 1110 **C.** 11100 **D.** 0001 **E.** 1110000000

e. Keyboard codes

Use the number pad on the keyboard. Whilst holding down the ALT key, type 65. The letter **A** should appear on the screen. Now repeat but his time, enter 66, then 67 and so on. The letters **B** followed by **C** followed by **D** should appear on your screen. The Table below shows the decimal number and the corresponding letter for codes 65-72.

Decimal Number	Letter
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H

f. Encryption

A Vigenère cipher wheel.

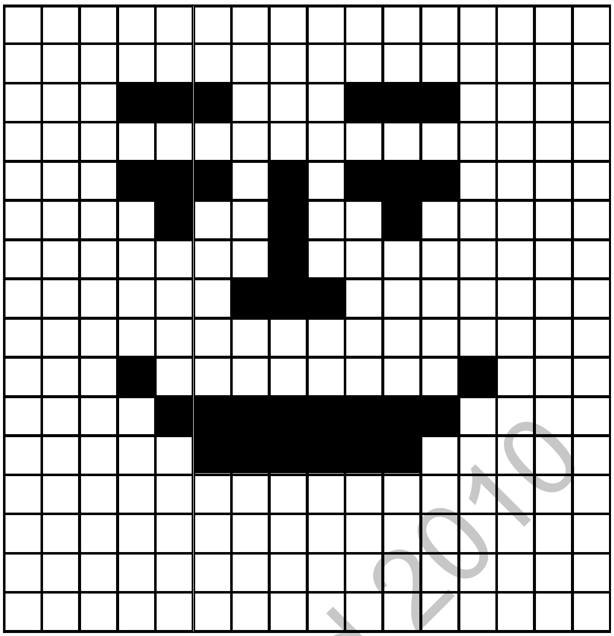
Can you decode ZWDH EW?

g. Images and pixels

- Everything in a computer is just numbers: even pictures
- A digital camera converts a picture that it has taken ("encodes it") to a long list of numbers.
- When the digital camera displays it on the screen it is converting the numbers back into a picture ("decoding" them)
- A simple way is to put a fine grid over the picture, and store a number corresponding to the colour of each square.
- With black and white pictures, 0 could be used for white and 1 for black.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	1	0	1	1	1	0	0	0
0	0	0	0	1	0	0	1	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

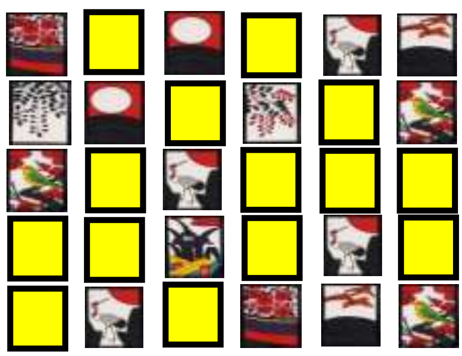
- To get the picture back from the 1s and 0s (decode it), you just go through the grid, filling in the squares corresponding to 1s and leaving the 0's.

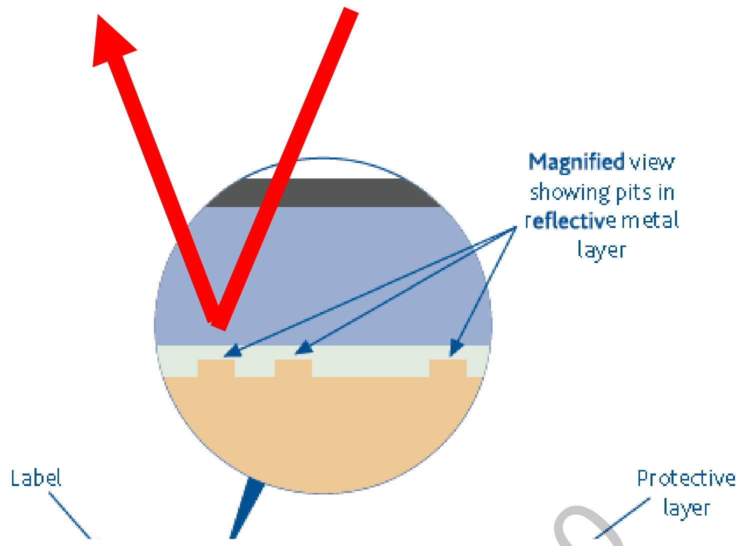


- Computer screens are divided into pixels
- Pixels are small dots that make up an image
- Stored images are shown to us using these pixels
- Sometimes if you enlarge an image you can see the different pixels
-

3. Error correction

See the card trick to discover how computers can correct errors.





4. Wii controller

The two most hard working students get to go first on Wii tennis game.