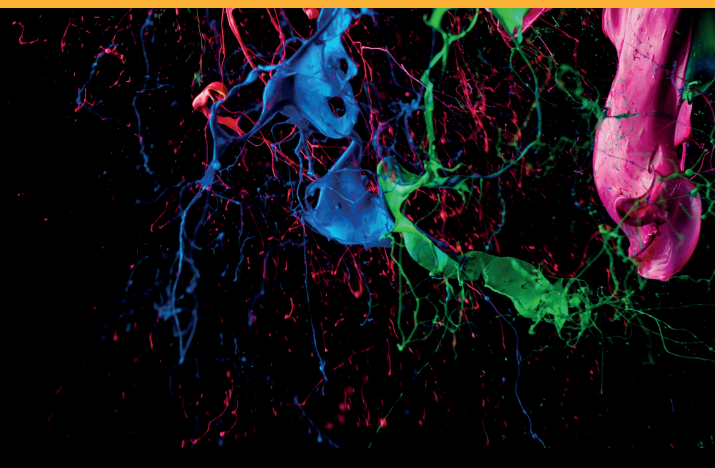
CAMBRIDGE

Brighter Thinking

GCSE for OCR

COMPUTER SCIENCE

Teacher's Resource



CAMBRIDGE

Brighter Thinking

GCSE for OCR

Teacher's Resource

Shahneila Saeed and David Waller

Course Consultant: Ann Weidmann

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org

Information on this title:

www.cambridge.org/9781316504093 (Elevate edition)

www.cambridge.org/9781316504109 (Free online)

© Cambridge University Press 2016

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2016

A catalogue record for this publication is available from the British Library

ISBN 978-1-316-50403-3 Elevate edition

ISBN 978-1-316-50410-9 Free online

Additional resources for this publication at **www.cambridge.org/education**

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

NOTICE TO TEACHERS IN THE UK

The photocopy masters in this publication may be photocopied or distributed [electronically] free of charge for classroom use only. Worksheets and copies of them remain in the copyright of Cambridge University Press.

Contents

© Cambridge University Press

Introduction

This book has been written to support you in delivering the **OCR J276 GCSE Computer Science** specification. It accompanies Cambridge University Press's GCSE Computer Science for OCR student textual and online resources.

The structure of the Teacher's Resource closely matches the Student Book which is divided into chapters and sections covering the specification content. The Teacher's Resource contains a chapter of teaching guidance for each Student Book chapter.

An introduction details **learning outcomes, what your students need to know** in terms of prior learning in order to tackle the contents of the chapter and key vocabulary that you should introduce to them. The introduction also describes common misconceptions, suggesting how to address them, and offers 'hooks' to introduce the topics in an engaging way.

A **skills and coding** section lists the **maths** and **coding** skills that are covered in the chapter.

The **Skills and coding for non-specialist teachers** section covers the programming and coding introduced in each chapter in greater depth in addition to explaining specialist concepts such as decomposition and abstraction.

The Teacher's Resource also provides:

- **prompting questions**, to promote discussion of the topic
- suggested activities for
  - **starters,**
  - **plenaries,**
  - **enrichment** and
  - **assessment** for each chapter of the Student Book
- full solutions and answers for all the chapter activities.

Answers to the Practice Questions can be found in the GCSE Computer Science for OCR Cambridge Elevate enhanced edition.

A scheme of work is included with learning outcomes, suggested teaching times and showing how the chapter topics cover the items in the specification.

Advice is also provided for the **Non-Exam Assessment (NEA)** component.

© Cambridge University Press

Changes to GCSE Computer Science

This chapter offers an overview of the main changes to the GCSE computer science qualification for first teaching from September 2016.

Introduction

The OCR computer science qualification contains the **GCSE Subject Level Conditions and Requirements for Computer Science** (Ofqual/15/5681), published by Ofqual in May 2015.

This can be downloaded from: **https://www.gov.uk/government/publications/gcse-9-to-1-subject-level-conditions-and-requirements-for-computer-science**

The document stipulates the assessment objectives, the nature of the non-examination assessment and the subject aims and content that must be incorporated by all examination boards.

Grades

The new GCSE is significantly different from its predecessor. Students will now be graded on a 1–9 scale. If you wish to read more about grades, the latest information can usually be found on the websites of Ofqual and the awarding bodies.

Assessment

The GCSE (9–1) in Computer Science is a linear qualification with a 100% terminal rule.

There are three components, two externally examined components (01 and 02) weighted at 40% each and a non-exam assessment (03) weighted at 20% that is assessed by the centre and externally moderated by OCR.

Each examined component consists of an exam paper with a duration of 1 hour 30 minutes. The non-exam assessment has a duration totalling 20 hours. Students must take all three components.

There will be one examination series available each year in May/June to **all** students and all examined components must be taken in the same examination series at the end of the course.

Students will be able to retake the examination as many times as they wish and can choose either to retake the non-exam component or to carry forward their mark from their previous sitting.

The assessment objectives and their weightings are given in the following table:

	Assessment Objective	Weighting
AO1	Demonstrate knowledge and understanding of the key concepts and principles of Computer Science.	30%
AO2	Apply knowledge and understanding of key concepts and principles of Computer Science.	40%

© Cambridge University Press

	Assessment Objective	Weighting
AO3	Analyse problems in computational terms: to make reasoned judgements to design, program, evaluate and refine solutions.	30%

Subject content

The following tables shows the changes in subject content between the new OCR J276 Computer Science specification and the J275 Computing specification which it replaces.

Learning objectives in J275 that are not in J276	
Specification reference	**Statement**
2.1.1 (a)	define a computer system
2.1.1 (c)	explain the need for reliability in computer systems
2.1.1 (d)	explain the need for adherence to suitable professional standards in the development, use and maintenance of computer systems
2.1.2 (j)	explain how the amount of RAM in a personal computer affects the performance of the computer
2.1.2 (n)	discuss how changes in memory technologies are leading to innovative computer designs
2.1.2 (p)	describe suitable input devices for a wide range of computer controlled situations
2.1.2 (q)	describe suitable output devices for a wide range of computer controlled situations
2.1.2 (r)	discuss input and output devices for users with specific needs
2.1.4 (q)	explain how the computer distinguishes between instructions and data
2.1.5 (a)	describe a database as a persistent organised store of data
2.1.5 (b)	explain the use of data handling software to create, maintain and interrogate a database
2.1.6 (c)	describe how a DBMS allows the separation of data from applications and why this is desirable
2.1.5 (d)	describe the principal features of a DBMS and how they can be used to create customised data handling applications
2.1.5 (e)	understand the relationship between entities and tables
2.1.5 (h)	explain the use of key fields to connect tables and avoid data redundancy
2.1.6 (a)	explain the advantages of networking stand-alone computers into a local area network
2.1.6 (h)	describe and justify network policies such as acceptable use, disaster recovery, failover, back up, archiving
2.1.6 (l)	explain the importance of HTML and its derivatives as a standard for the creation of web pages
2.1.6 (m)	describe common file standards associated with the internet such as JPG, GIF, PDF, MP3, MPEG

© Cambridge University Press

Learning objectives in J276 that were not in J275	
Specification reference	**Statement**
1.1	Von Neumann architecture: MAR (Memory Address Register) MDR (Memory Data Register) Program Counter Accumulator
1.1	common CPU components and their function: ALU (Arithmetic Logic Unit) CU (Control Unit) Cache
1.1	embedded systems: purpose of embedded systems examples of embedded systems
1.4	factors that affect the performance of networks
1.5	Wi-Fi: frequency and channels Ethernet
1.5	the uses of IP addressing, MAC addressing, and protocols including: TCP/IP (Transmission Control Protocol/Internet Protocol) HTTP (Hyper Text Transfer Protocol) HTTPS (Hyper Text Transfer Protocol Secure) FTP (File Transfer Protocol) POP (Post Office Protocol) IMAP (Internet Message Access Protocol) SMTP (Simple Mail Transfer Protocol)
1.5	the concept of layers packet switching
1.6	forms of attack
1.6	threats posed to networks: malware phishing people as the 'weak point' in secure systems (social engineering) brute force attacks denial of service attacks data interception and theft the concept of SQL injection poor network policy

© Cambridge University Press

Learning objectives in J276 that were not in J275	
1.6	Identifying and preventing vulnerabilities: penetration testing network forensics network policies anti-malware software firewalls user access levels passwords encryption
1.8	how key stakeholders are affected by technologies
1.8	cultural implications of Computer Science
1.8	legislation relevant to Computer Science: The Data Protection Act 1998 Computer Misuse Act 1990 Copyright Designs and Patents Act 1988 Creative Commons Licensing Freedom of Information Act 2000
2.1	computational thinking: abstraction decomposition algorithmic thinking
2.1	standard searching algorithms: binary search linear search standard sorting algorithms: bubble sort merge sort insertion sort
2.2	the use of basic file handling operations: open read write close
2.2	the use of records to store data
2.2	the use of SQL to search for data
2.2	how to use sub programs (functions and procedures) to produce structured code
2.3	defensive design considerations: input sanitisation/validation planning for contingencies anticipating misuse authentication
2.3	maintainability: comments indentation
2.3	the purpose of testing

© Cambridge University Press

Learning objectives in J276 that were not in J275	
2.3	types of testing: iterative final/terminal
2.6	binary shifts
2.6	check digits

Non-exam assessment

In J276 there is only one non-exam assessment component, the **Programming Project**.

There is no Practical Investigation that was present in J275.

The following additional programming techniques and requirements for the programming project are new in J276:

- how to understand and use basic file handling operations:
  - open
  - read
  - write
  - close
- how to analyse and identify the requirements for a solution to the problem
- how to set clear objectives that show an awareness of the need for real world utility
- how to use abstraction and decomposition to design the solution to a problem
- how to identify the data requirements for their system
- how to design suitable input and output formats and navigation methods for their system
- how to use functions/sub programs to produce structured reusable code.

© Cambridge University Press

Chapter 1: Algorithms

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain what an algorithm is and create algorithms to solve specific problems
- use sequence, selection and iteration in algorithms
- use input, processing and output in algorithms
- express algorithms using flow diagrams and pseudocode
- analyse, assess and compare different algorithms
- create, name and use suitable variables
- use arithmetic, relational and Boolean operators
- use conditional statements.

What your students need to know

No prior knowledge is expected for this chapter.

Vocabulary

- Algorithm
- Sequence
- Selection
- Iteration
- Input, output and processing
- Flow diagram
- Pseudocode
- Variable
- Identifier
- Constant
- Arithmetic operators
- Relational operators
- Boolean (logical) operators
- Nested operations

Common misconceptions and other issues

- Students should be encouraged to use the formal conventions of creating flow diagrams and using pseudocode.
- Students should use meaningful variable identifiers (for example, 'Age' for a variable about age rather than 'X'), also use indentation and commenting in their pseudocode. This will make code easier to read and check for errors.
- Students will probably be unfamiliar with the arithmetic operators MOD and DIV. Give examples similar to those shown in the arithmetic operators table.
- The relational operators == and != will also need explanation as using '=' instead of '==' is a common syntax error. ('=' is for 'assignment', whereas '==' is for comparison'.)
- The Boolean operators 'AND' and 'OR' can cause confusion as the statement 'I would like red ones and blue ones' would require the 'OR' operator when selecting from a list or array.
- When using nested selection, care must be taken in completing each block with an 'endif' statement.

© Cambridge University Press

Skills and coding

- Maths skills:
  - Arithmetic operators
  - Order of operations (BIDMAS)
  - Calculation of average.
- Coding skills:
  - Use of pseudocode
  - Declaring and assigning variables
  - Selecting suitable identifiers
  - Selection using 'if…then…elseif…else' statements
  - Nested 'IF' statements
  - Use of 'switch/case' statements.

Skills and coding for non-specialist teachers

1. Use of pseudocode

The pseudocode commands and key words are given in the OCR Pseudocode Guide.

The guide states:

The following guide shows the format pseudocode will appear in the examined components. It is provided to allow you to give learners familiarity before the exam. Learners are not expected to memorise the syntax of this pseudocode and, when asked, may provide answers in any style of pseudocode they choose providing its meaning could be reasonably inferred by a competent programmer.

The pseudocode is to provide a format in which students can express the logic of an algorithm and the statement stresses that they will not be penalised if their answers do not exactly match the conventions shown in the guide, that is, syntax errors will not be penalised.

2. Declaring and assigning variables and selecting suitable identifiers

A variable is defined as a 'named container' for a value that can change as a program is running.

Values are assigned using the '=' symbol.

Variables should be given meaningful names (identifiers) and the naming convention should be consistent, for example: firstName or first_name or FIRST_NAME.

3. Selection using 'if…then…elseif…else' statements

These are explained in the section 'Relational operators' and the solutions to **Activities 1.9 and 1.10**.

They are used to check variable values where there is one or multiple alternative responses.

'endif' should always be used to close the statement.

For example:

```
if index == 3 then
        print("The variable index is equal to 3")
endif
```

'==' is used to check equality.

If there are two alternative actions then 'else' can be used.

© Cambridge University Press

For example:

```
If index == 3 then
        print("The variable index is equal to 3")
else
        print("The variable index is not equal to 3")
endif
```

The 'elseif' statement is also used in the book.

For example:

```
If index == 3 then
        print("The variable index is equal to 3")
elseif  index > 3 then
        print("The variable index is greater than 3")
else
        print("The variable index is less than 3")
endif
```

4. Nested 'if'? statements.

The last two examples are often referred to as 'nested if' statements as they contain more than one selection and action.

However, 'nested if' also refers to two distinct 'if' statements, one inside the other.

This is explained through a worked example.

When using these two distinct statements, care must be taken with indentation and also that an 'endif' is used for each one.

For example, this section of code would calculate a customer discount but would then limit it to a maximum amount of £20. It then calculates the final price allowing for the discount.

```
if price >= 100 then
        discount = price/100*10
        if discount > 20 then
                discount = 20
        endif
        price = price – discount
        print(price)
endif
```

1. Use of 'switch/case' statements.

This is another selection method and it is useful where there are many alternative responses based upon a single variable value, that is, no Boolean operators are used.

It is especially useful when evaluating user input, although the selection could also be accomplished using 'if…then…elseif…else) statements.

This is explained through a worked example and the solution to **Activity 1.12**.

© Cambridge University Press

Prompting questions

- What is an algorithm? Have you heard this term before? For which subject, in which context?
- Can you think of any algorithms that you follow in your everyday life?
- Can you think of any activities where you use selection or iteration?

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries:

- The human robot: Ask students to consider a routine that is part of their daily life. Can they write down a set of instructions that will allow someone else to follow their routine completely? A useful extension to this activity is to ask students to pair up and role play their partner's instructions. The students can then provide each other with feedback and discuss the quality of those instructions and what can make them better. This is a simple way to enable students to begin writing and testing their own algorithms.
- Divide the class into pairs. One partner sees an image and is going to be the programmer, the other is going to be the human computer and will carry out the instructions they receive. It is important that the human computer doesn't see the image being used by the programmer. The activity is simple; the programmer describes the image they see in front of them and the human computer tries to recreate it. This can be done using simple paper and pen or by getting creative and using playdough instead!

Enrichment activities:

- There is magic in computer science. The following resource **www.cambridge.org/links/kotd4001** uses magic tricks to explain algorithms and many other computer science concepts. Choose a trick that works for you as a teacher and demonstrate it to the students. Challenge the students to see if they can figure out how the trick works. Encourage them to write the algorithm for it as best they can, refining it as they go and making use of the concepts learnt.
- Set the students a homework task or a challenge to write an article using inspiration from the CS4N style to explain a concept you have learnt in this chapter.

Assessment ideas:

- Set the students a challenge using the following scenario: Write an algorithm for a Joke Generator program. When the user runs the program, it begins by displaying a joke on screen. The user is then asked if they would like another joke, if the user answers 'yes' then another joke appears on the screen. This version of the program makes use of simple selection and sequence statements. Nested IF's and select/case statements can be added if the user is asked to input a number between 1 and 10. Once entered, the program will display a joke which corresponds to that number. Ask students to submit the following:
  - A flow diagram that explains how their program should work
  - Annotated pseudocode of the algorithm.

© Cambridge University Press

Answers

Activity 1.1

Students will write this as a list of steps, they may be in any format and some might put this into a diagrammatic format. There is no single right answer to this activity and this is important to emphasise. The key thing is to check the logic at this stage, does it work? Encourage the students to mime their algorithm to try it out. Often students will forget to write in when to stop pouring the water or milk, for example. The activity emphasises the need for instructions to be specific and precise, concisely written and in a logical order.

An example of how the algorithm might look is:

- Fill kettle with cold water to maximum level
- Check to see if kettle is plugged in
  - If it isn't plugged in then plug it in
  - Check to see kettle is switched on
  - Switch it on if it isn't
- Is kettle boiled?
  - If No, then wait until the kettle has boiled
- Put 1 teabag into cup
- Pour boiled water into cup until full
- Leave for 1 minute
- Stir for 20 seconds
- Remove teabag and place in bin
- Is milk required?
  - If No, then end
  - If Yes, then pour in 10 ml of milk
- Is sugar required?
  - If No, then end
  - If Yes, then add required sugar

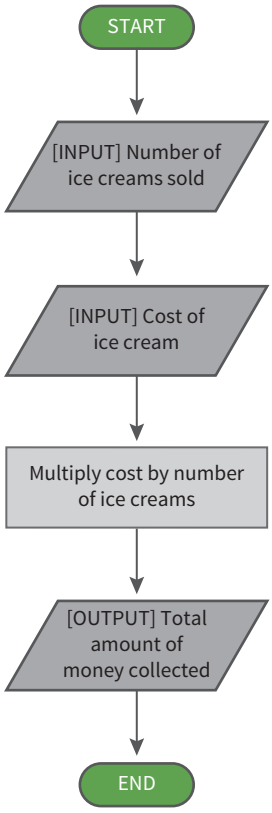
Activity 1.2

There may be multiple variations in which the answer to this activity might be presented. The key thing is the logic and precision in the instructions as well as the correct labelling of instructions as sequence, selection or iteration.

- Put plug in the bath //Sequence
- Turn on hot tap //Sequence
- Is the water at the correct temperature? //Selection
- Is it too hot? //Selection
- Turn cold tap until water is at the correct temperature //Iteration
- Is the water at the correct temperature? //Selection
- Is it too cold? //Selection
- Turn off cold tap until water is at the correct temperature. //Iteration
- Is there enough water in the bath? //Selection
- Turn off both taps. //Sequence

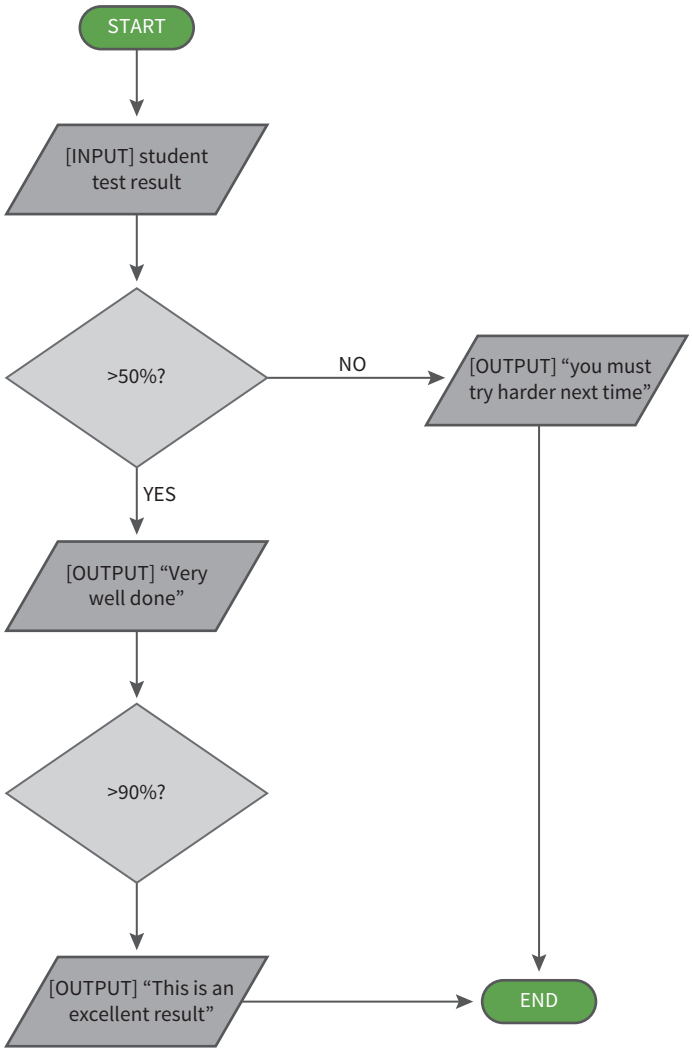
© Cambridge University Press

Activity 1.3



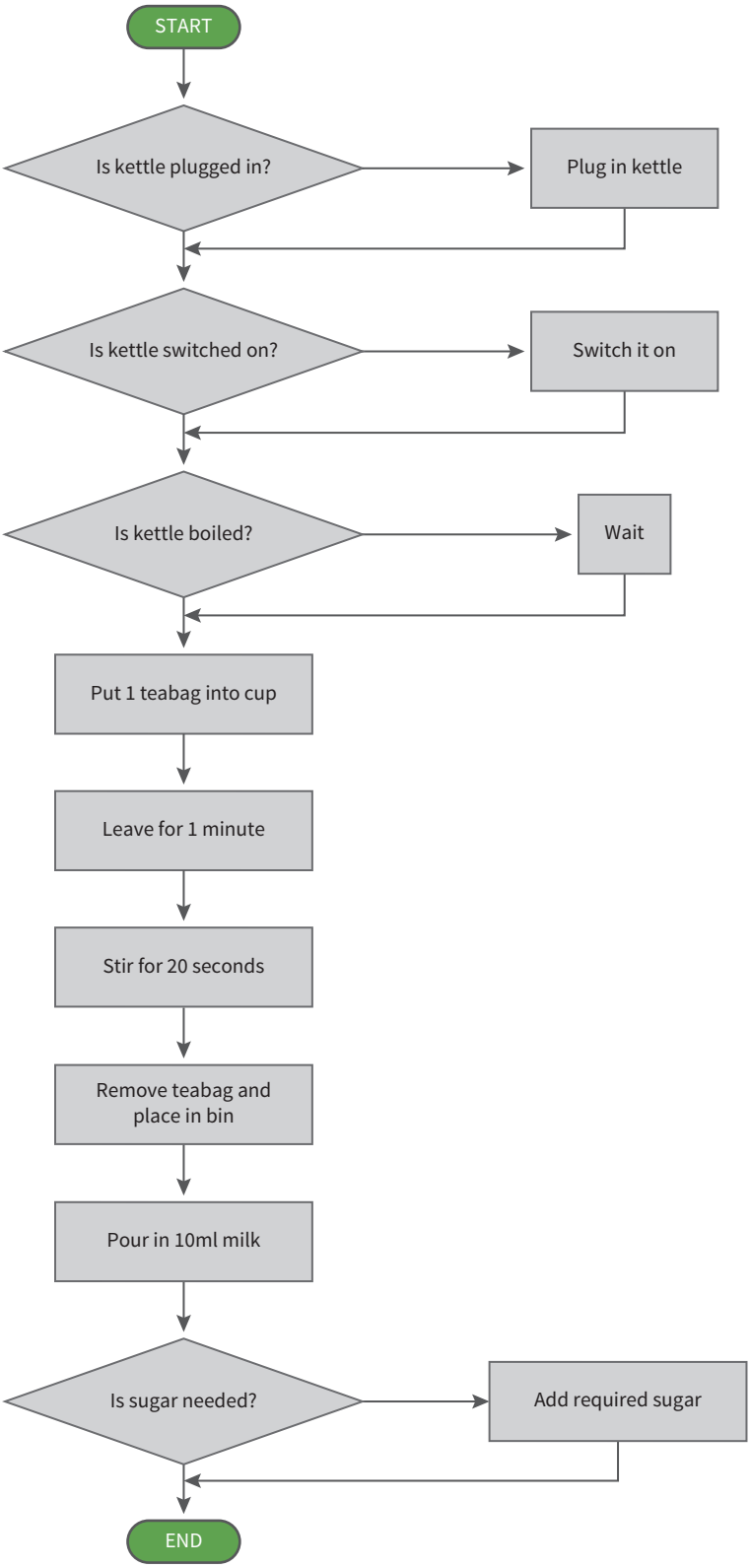
© Cambridge University Press

Activity 1.4



© Cambridge University Press

Activity 1.5



© Cambridge University Press

Activity 1.6

The variables used are: Username, Year, First, Last, X

- Catherine Jones 2005 becomes '05CJones1'
- Fred Green 2006 becomes '06FGreen1'

Username '03SSmith13' tells us the following:

- The user joined in 2003
- Their first name begins with S
- Their surname is Smith
- There are 13 users with the surname Smith

Activity 1.7

```
Name = input ("Please enter your name")          //Ask user for their name
Age = input ("Please enter your age")            //Ask user for their age
Print ("Hello", Name, ". You are", Age, "years of age")   //Prints personalised msg.
```

Activity 1.8

```
diameter = input ("Please enter diameter of wheel")   //User enters diameter
radius = diameter/2                                   //Calculate radius as half of a diameter
Pi = 3.142
area = Pi * radius ^2                                 //Area calculated by Pi multiplied by radius squared
print ("The area calculated is: " + area)            //Prints personalised msg
```

Activity 1.9

```
number = input ("Please enter a number between 1 and 10")    //User enters number
if number <= 5 then
        print (number + " is a low number.")
else
        print (number +  " is a high number.")
End IF
```

Activity 1.10

The comment generated would have been "You have gained half marks."

The algorithm checks first to see if a number is less than 5. After that, the first case is to check to see if the value is greater than 5. If it is, the message is output and the algorithm ends. Nine is greater than five so it would not have progressed beyond that case.

© Cambridge University Press

Activity 1.11

```
FirstName = input ("Please enter the first name")
Surname = input ("Please enter the surname")
Year = input ("Please enter the year group")
if (Year >=7 AND Year <=11) then
    print ("This is a correct year group")
else
    print ("This is an incorrect year group")
endif
TutorGrp = input ("Please enter the tutor group")
if (TutorGrp=="red" OR TutorGrp=="blue" OR TutorGrp=="green" OR TutorGrp=="yellow") then
    print ("This is a suitable tutor group")
else
    print ("This is an incorrect tutor group")
endif
```

Activity 1.12

```
MonthNo = input ("Please enter the number of the month")
switch MonthNo:
    case 1:
        print ("January")
    case 2:
        print ("February")
    case 3:
        print( "March")
    case 4:
        print ("April")
    case 5:
        print ("May")
    case 6:
        print ("June")
    case 7:
        print ("July")
    case 8:
        print ("August")
    case 9:
        print ("September")
    case 10:
        print ("October")
```

© Cambridge University Press

```
case 11:
    print ("November")
case 12:
    print ("December")
default:
    print("Entry not recognised")
endswitch
```


© Cambridge University Press

Chapter 2: Iteration

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain what is meant by iteration
- explain the difference between definite and indefinite iteration
- use for loops
- use while loops
- use do…until loops
- use nested loops
- analyse algorithms using trace tables
- use iteration when designing algorithms.

What your students need to know

Students should:

- be able to create flow diagrams using the standard symbols
- be able to use pseudocode to display algorithms
- be familiar with the OCR Pseudocode Guide.

Vocabulary

- Iteration
- Definite iteration
- Indefinite iteration
- For loop
- While loop
- Do…until loop
- Nested loop
- Infinite loops
- Trace table
- Algorithm efficiency
- Logical error

Common misconceptions and other issues

When using pseudocode, the students should be encouraged to use both comments and indentation. This will make code easier to read and check for errors.

A common error when coding a 'while' loop is not to declare the variable and assign a value before the loop starts as the condition is checked at the start of the loop.

For example,

```
while password = ""
        password = input("Please enter the password.")
        if password != "Password!" then
                password = ""
        endif
```

© Cambridge University Press

```
endwhile
```

This would generate an error as the variable 'password' needs to be declared and assigned the value of an empty string before the 'while' loop starts, that is:

```
password = ""
while password = ""
        password = input("Please enter the password.")
```

etc.

This is not necessary in a 'do…until' loop, as the condition is checked at the end of the loop.

For example:

```
do
        password = input("Please enter the password.")
until password == "Password!"
```

Students should check the conditions of the loop as it is very easy to start an infinite loop (a loop that can never achieve the condition to stop it).

In the OCR pseudocode, there is no command for generating a random number. Students should be encouraged to investigate the method in the language they are studying and use that or the method used in the book.

When creating nested loops, care must be taken to close each loop using the 'endwhile' or 'next' statements.

Skills and coding

- Maths skills:
  - Times tables (required especially for **Activity 2.4**)
- Coding skills:
  - Use of pseudocode
  - Declaring and assigning variables
  - Selecting suitable identifiers
  - for…next loops
  - while loops
  - do…until loops
  - User input.

The activities will also reinforce the skills acquired in **Chapter 1**.

Skills and coding for non-specialist teachers

1. Definite iteration

This is used in loops where the number of iterations is known before the loop starts.

They can be created using 'for…next' and 'while…endwhile' statements.

To use a 'for…next' loop to iterate 10 times, the following structure should be used:

```
for counter = 0 to 9
        print(counter)
next counter
```

The loop could be set from 1 to 10 but using 0 to 9 reinforces the idea that computers start counting at 0 rather than 1.

A suitable variable name should be used (so, 'for counter', as used here, is better than, 'for x').

When using a 'while' loop the variable should be declared and assigned a value before the start of the loop.

© Cambridge University Press

```
counter = 0
while counter <= 9
        print(counter)
        counter = counter +1
endwhile
```

This will produce the same output as the 'for…next' loop above.

Definite iteration is explained in the Student Book and the solution to **Activity 2.1**.

2. Indefinite iteration

In indefinite iteration the number of iterations is not known before the loop is started.

The iterations stop when a certain condition becomes true or false.

The 'while…endwhile' and 'do…until' loops from the pseudocode can be used for this iteration.

A 'while' loop continues *while* a certain condition remains true.

The condition is checked *before* the code is executed and so the variable used must be assigned the value to make the loop run.

```
answer = 'no'
while answer == 'no'
        answer = input('Please enter 'yes' or 'no')
endwhile
```

This loop will continue until the user entry is not 'no'. It doesn't have to be 'yes' anything but 'no' will stop the loop.

A 'do…until' loop is similar to a 'while' loop but the comparison is not done until the end of the code block and the loop will run while the condition remains false.

```
do
        answer = input('Please enter 'yes' or 'no')
until answer == 'no'
```

If the user input is 'no' then the loop will stop but with any other input, the loop will continue to run.

Indefinite iteration is explained in the Student Book and the solutions to **Activities 2.2 and 2.3**.

3. Nested loops

A nested loop is a loop that runs inside another loop.

Care must be taken that each loop is terminated correctly.

```
for index = 1 to 3
        for count = 1 to 10
                print(count)
        next count
next index
```

This nested loop would output 1 to 10 three times as for each turn of the outer loop, the inner loop would run ten times.

Nested loops are explained in the Student Book and the solution to **Activity 2.4**.

© Cambridge University Press

4. Trace tables

When creating trace tables, all variables, inputs and outputs should be given column headings.

Entries should be made in the rows to correspond to the values of the variables, inputs and outputs.

```
index = 1
x = 0
y = 0
do
        x = index * index
        if x >9 then
                y = x * 3
                print(y)
        endif
        index = index + 1
until index > 5
```

This can be shown in the following table:

index	x	y	output	Comments
1	0	0		At the start, this is the state of the variables.
1	1	0		At the first iteration of the do…until loop x = 1*1 = 1 and therefore as x is less than 10, y remains at 0 and there is no output.
2	4	0		The 'index + 1' command means that index = 2 and x = 4. This is still less than 10.
3	9	0		x is not greater than 9 and so y remains at 0.
4	16	48	48	x is now greater than 9 and so y = 3 times the value of x and its value is output.
5	25	75	75	

Obviously, in an actual trace table, the comments column would not be included.

As this is a do…until loop, index is evaluated until, and including when, index is equal to 5.

It will stop when index becomes equal to 6.

Trace tables are explained in the Student Book and the solution to **Activity 2.5** and in the practice questions.

Prompting questions

- Can you think of any everyday examples where iteration comes into play?
- Can you think of any computer games that you have played that make use of iteration?
- Why is iteration important/useful? How does it help programmers?

© Cambridge University Press

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- Ask students to consider one of the algorithms they wrote for **Chapter 1**. Can they modify their flow diagram and/or pseudocode to include iteration? As an early activity in this chapter, iteration can be illustrated simply by using the word 'repeat' in pseudocode or a looping arrow on the flow diagram. If the activity is used as a plenary, then you can use this task to assess students' understanding of using the appropriate type of iteration.

Examine a computer game (*note: either let students select their own or pick one for them. The game selected should include some form of iteration and this might be visible in the form of scoring or gaining/losing lives)*. Ask students to see if they can find an example of iteration within the game and ask them to explain the reasons behind their decision. This activity can be extended by asking students to:

  - write the algorithm for the example which you have found, either as pseudocode or as a flow diagram (or both)
  - explain what type of iteration do you think is being used.

- Play a game of Word Sneak using words from the 'programming deck'. The full set of resources and activity guides can be downloaded from: **www.cambridge.org/links/kotd4002** Word Sneak is a fun activity to help you assess students' knowledge and understanding in a unique way. The aim of the game is to have a normal conversation and sneak your hidden words into what you say without your partner noticing. The first person to use all their hidden words is the winner.
  - A variation of this is Three Word Stories (**www.cambridge.org/links/kotd4003**) that requires the player to engineer the story being told (three words at a time) so that their partner uses their hidden word.

Enrichment activities

- Provide students with an algorithm written using only sequential instructions. Ask them to modify the code to adapt it to include the appropriate iteration in the right places. Does the code work?

Assessment ideas

- Ask students to devise a quick game consisting of a single level. Students will need to explain what the game does as well as write the pseudocode for the main game functions. Tell students that their pseudocode should include examples of sequence, selection and iteration.

Answers

Activity 2.1

```
index = 0

number = input("Please enter a number:")

while index <=12

    print(index + " x " + number + " = " + number*index)

    index = index + 1

endwhile
```

© Cambridge University Press

Activity 2.2

```
sum = 0
another = "y"          //This will determine if another number is to be entered
while another == "Y"
        number = input("Please enter a number.")
        sum = sum + number
        another = input("Do you want to enter another number (y/n)")
endwhile
print("The sum of the numbers is " + sum)
```

Activity 2.3

```
play = "y"                                          //This variable determines if the game should run again
while play == "y"
        mysteryNumber = a random number between 1 and 100
        guess = 0
        while guess == 0
                guess = input("Please enter a number between 1 and 100.")
                if guess > mysteryNumber then
                        guess = 0
                                print("Your guess is too high.")
                        elseif guess < mysteryNumber
                                guess = 0
                                print("Your guess is too low.")
                endif
        endwhile
        print("Well done. You guessed correctly!")
        play = input("Do you want to play again (y/n)")    //The user is asked if they want to play again.
endwhile
print("Thank you for playing the game. Hope you enjoyed it.") //The user is given a departing message if they do not want to play again.
```


© Cambridge University Press

Activity 2.4

- The algorithm asks the user to enter the upper and lower limits of the range. These are then stored in two variables. The variables are then used within the loop to define the iterations.

```
LowerRange = input("Enter the number where you would like your range of tables to begin")

UpperRange = input("Enter the number for the upper limit for the range of tables")

for index = LowerRange to UpperRange
        print ("this is the" + index + " times table")
    for times = 2 to 12
            print (times + "x" + index + "=" + index*times)
     next times
next index
```

Activity 2.5

turns	x	output
0	3	
0	9	
3	27	
6	81	
9	243	
12	729	
15	2187	
18	6561	
21	19683	19683

© Cambridge University Press

Chapter 3: Boolean logic

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- create truth tables for Boolean operators
- draw AND, OR and NOT logic gates
- combine logic gates into logic circuits
- create truth tables for logic circuits.

What your students need to know

Students should:

- have knowledge of Boolean or logical operators
- be able to use and understand pseudocode representations using logical operators.

Vocabulary

- Logical operator
- Boolean logic
- Transistor
- Truth table
- Logic gate
- AND gate
- OR gate
- NOT gate
- Logic circuit
- Control system

Common misconceptions and other issues

Care should be taken when formulating and evaluating NOT statements.

For example, Q = NOT(A AND B) is not the same as Q = NOT(A) AND NOT(B).

A truth table for Q = NOT(A AND B) would be:

A	B	Q
T	T	F
T	F	T
F	T	T
F	F	T

The outcome is the reverse of a truth table for an AND statement. Any situation where both A AND B are not true would cause Q to be true.

© Cambridge University Press

A truth table for Q=NOT(A) AND NOT(B) would be:

A	B	Q
T	T	F
F	T	F
T	F	F
F	F	T

This statement requires both A and B to be false for Q to be true.

Skills and coding

- Coding skills:
  - Use of pseudocode
  - Using and interpreting logical operators in pseudocode.

Skills and coding for non-specialist teachers

1. Truth tables

Truth tables can be used to check the logic of all statements using logical operators from simple statements such as:

```
If password == "Password!" then
```

where the outcome can be assessed depending on whether the statement is true or false, to compound statements such as:

```
If(colour == "red" OR colour == "blue") AND size == "M" AND distance <= 10 then
```

where a table such as the one below could be used.

red	blue	M	<=10	Result
T	T	T	T	T
T	F	T	T	T
T	F	F	T	F
T	F	T	F	F
F	T	T	T	T
F	T	F	T	F
F	T	T	F	F

2. NOT statement

The NOT statement reverses the logic of AND and OR operators.

This is explained before **Activity 3.1** and reinforced in **Activity 3.1**.

In this activity, the statement to be evaluated is:

```
if NOT (X == 3 OR Y == 6) then
        print("Conditions are met.")
endif
```

© Cambridge University Press

Therefore, if either of the conditions is true, that is, if X is equal to 3, OR Y is equal to 6, then the compound statement is false.

Logic without NOT		
X	**Y**	**X OR Y**
T	T	T
F	T	T
T	F	T
F	F	F

Logic with NOT		
X	**Y**	**X OR Y**
T	T	F
F	T	F
T	F	F
F	F	T

3. Logic gates

Logic gates are built of transistors to electronically represent Boolean logic.

When using truth tables for logic gates and logic circuits, in which they are combined, true and false should be represented by 1 and 0 to signify whether there is or is not an input or output.

Prompting questions

- What's the difference between AND/OR/NOT?
- Can students come up with any examples of AND/OR/NOT?
- One example of AND/OR/NOT in everyday life might be "We are going to the circus if = the day is Saturday OR Sunday AND the circus is in town". Can students come up with any other examples of their own?
- What is a compound statement?
- Why are truth tables important? What do they help us do?

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- Ask students to write down three examples of situations where they see Boolean logic in action. These can then be shared with a partner before feeding back to the group.
- We can often see Boolean logic in games. For example, we only want the score to double if: the character has collected the object AND the object is a diamond. Ask students to consider another example of Boolean logic within a game and to write a truth table for it.

Enrichment activities

- Ask students to go through the algorithms for the game they designed in **Chapter 2**. Identify where Boolean logic applies and draw out a truth table for each one.
- Choose one of the algorithms from the previous enrichment activity and use it to draw out the logic gate for the truth table.

© Cambridge University Press

Assessment ideas

Provide students with a scenario for which they should attempt to draw out a truth table and an associated logic gate. Some examples of possible scenarios are:

- A leisure club will charge the concession rate if visitors are under 16 or over 65.
- A school's electronic security gates will open only if: it is a weekday and the ID card is a valid student card OR it is an administrator ID card.

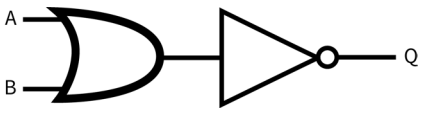
Answers

Activity 3.1

X	Y	Q
F	F	T
T	F	F
F	T	F
T	T	F

Activity 3.2

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0



© Cambridge University Press

Chapter 4: Data types and structures

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain what is meant by 'data type' and list some common types
- use the correct data types in algorithms
- carry out various manipulations such as finding the length of and slicing and concatenating 'string' data types
- create and work with simple array data structures
- create and work with two-dimensional arrays
- describe other data structures.

What your students need to know

Students should:

- be able to use pseudocode to create variables and display algorithms
- use suitable and consistent identifiers for variables
- understand the use of 'if…then…else' statements
- be able to use definite and indefinite iteration.

Vocabulary

- Data type
- Data structure
- Integer
- Real number
- Floating point number
- String
- String manipulation
- Index
- String traversal
- Concatenation
- Casting
- Array
- Two-dimensional array
- Cipher
- Encryption

Common misconceptions and other issues

In some programming languages, the data type of a variable does not have to be explicitly declared. Many of the languages that students are studying may be weakly-typed and therefore they might not have to declare a data type when creating variables.

If a variable is declared as:

```
index = 3
```

then it is being implicitly declared as an integer.

© Cambridge University Press

If it is declared as:

```
index = 3.33
```

then it is being implicitly declared as a float.

If it is declared as:

```
firstName = "Catherine"
```

then it is being implicitly declared as a string.

A common misunderstanding is when students are using the length of a string.

The number of characters or symbols in, or length of a string, is given by:

```
stringLength = string.length
```

The students should then use the value, stringLength – 1, when creating a loop to iterate through the string as index numbering starts at 0 and not 1.

Therefore, a string with a length of 10 will have characters indexed as 0 to 9.

If the loop was set up using stringLength instead of stringLength – 1, an error would be generated as there is no item with an index of 10.

For example:

```
string = "This is a string"
for index = 0 to string.length
    print(string(index))
next index
```

As the string length is 16, but there are only 15 index positions (0 to 15), an error message will be generated when the loop tries to access string(16).

When concatenating strings to form compound words, the students should remember to add spaces between the strings.

If it was concatenated in the following way:

```
firstName = "Jack"
lastName = "Smith"
fullName = firstName + lastName
```

The result would be 'JackSmith'.

A space should be included in the concatenation:

```
firstName = "Jack"
lastName = "Smith"
fullName = firstName + " " + lastName
```

The result would be 'Jack Smith'.

The definition of 'array' usually states that it contains a group of elements of the same data type but some programming languages allow different data types.

In the pseudocode, there is no command for finding the length of an array. In the book 'length' is used as that command is used for strings.

© Cambridge University Press

Skills and coding

- Maths skills:
  - Maximum, minimum and mean (required for **Activity 4.9** and **Activity 4.14**).
- Coding skills:
  - Use of pseudocode
  - Finding the length of a string
  - Creating a loop to traverse a string
  - Searching a string for a particular character
  - Counting the number of times that a character occurs in a string
  - Splitting strings
  - Finding substrings within a string
  - Concatenating strings
  - Casting
  - Creating, populating, editing and searching arrays
  - Creating, populating, editing and searching two-dimensional arrays.

Skills and coding for non-specialist teachers

1 Strings

The length of a string is found by using the 'length' command.

Therefore:

```
myString = "David"

lenString = myString.length
```

Would assign the value 5 to the variable lenString.

A loop to traverse the string would start at 0 (the first index position) and end at lenString-1 or index position 4.

```
for index = 0 to lenString – 1
    print(myString(index))
next index.
```

This would output:

```
D
a
v
i
d
```

Traversing a string is explained in a worked example in the Student Book and the solution to **Activity 4.2.**

2 Finding substrings

When traversing a string to find a substring, the length of the substring must be taken into account when creating the traversal loop.

For example:

```
myString = "David"
```

The length of the string is 5, and in the traversal above, the loop would start at 0 and end at 4 (lenString – 1).

© Cambridge University Press

But if the substring searched for was 'av' which has a length of 2, the search should end at index position 3 (lenString – 2).

This is explained in a worked example in the Student Book and the solution to **Activity 4.3**.

3 Arrays

Data items in an array have an index position in the same way as characters in a string.

Indexing starts at 0 and so when traversing an array, a loop should run to length of the array -1.

This is explained in the Student Book and the solutions to **Activities 4.6 to 4.8**.

4 Creating subarrays

A subarray can be created by copying data items from the main array using the indexes of the data items.

Activity 4.10 asks students to create a subarray using items that are not contiguous. The students are asked to traverse the main array to find items that are equal to or greater than 5 and then create a new array, called 'pass', to store them.

The algorithm should:

- traverse the array to find the total of all data items equal to or greater than five
- create a new array to store these items
- copy them to the new array.

A possible solution is given below.

```
array marks = [6, 9, 2, 5, 8, 3, 9, 9, 10, 9, 5, 7, 10]
count = 0
for index = 0 to marks.length-1
    if marks[index] >= 5 then
        count = count + 1
    endif
next index
array pass[count]
newIndex = 0
for index = 0 to marks.length-1
    if marks[index] >= 5 then
        pass[newIndex] = marks[index]
        newIndex = newIndex + 1
    endif
next index
```

© Cambridge University Press

5 Two-dimensional arrays

In a two-dimensional array, multiple data items can be stored at each index position. This makes the array more like a traditional database.

A two-dimensional array is declared as:

```
array myArray[6,2]
```

This stipulates that there will be six index positions and two items of data will be stored at each one.

More than two items of data can be stored at each index position:

```
array myArray[6, 3]
```

This would create a static two-dimensional array with three items of data at each index position.

If there are two items of data, they can be referenced as:

```
myArray[0, 0] and myArray[0, 1]
```

```
myArray[1, 0] and myArray[1, 1]
```

```
myArray[2, 0] and myArray[2, 1]
```

etc.

Prompting questions

- What is data?
- What is the difference between data and information?
- Look at the information around you (for example, on classroom walls, in text books, etc.) what are the different types of data that you can see? How would you categorise them?
- What are arrays?
- What are the different types of arrays, and when would you use each of them?
- What is a database?

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- Ask students to examine a resource, this can be anything from a poster to a document they have produced, a website, or simply going through their text book. What are the different types of information that they can see? Ask students to arrive at their own categories and ways of classifying the information. End the activity with groups sharing their work and explaining the reasons behind their categorisation. This will lead on nicely to students then being able to compare their work with the actual data types discussed later in the lesson.
- Play a game of Word Sneak using words from the 'data deck'. The full set of resources and activity guides can be downloaded from: **www.cambridge.org/links/kotd4004** Word Sneak is a fun activity to help you assess students' knowledge and understanding in a unique way. The aim of the game is to have a normal conversation and sneak your hidden words into what you say without your partner noticing. The first person to use all their hidden words is the winner.
  - A variation of this is Three Word Stories (**www.cambridge.org/links/kotd4005**) that requires the player to engineer the story being told (three words at a time) so that their partner uses their hidden word.
- Give students a list of different items of data, and for each one to correctly match the associated data type.

© Cambridge University Press

Enrichment activities

- Ask students to review the algorithms for their game design and to see if any would be more efficient if arrays would be used. Ask the students to make the adjustments to the code and then give the reasons for doing so.
- Investigate databases and SQL. There are many ways you can program in SQL and you don't always need PHP to do it. If your school has a database such as Microsoft Access you can create a new query and edit it in SQL view. This allows you to enter and execute SQL commands. Ask students to investigate this to create a single table of data storing a range of fields for a number of records. They should run SQL commands to allow them to add/delete/amend and search data within the records.

Assessment ideas

- Give students a range of topics, including: data types, static array, dynamic array, index, databases, SQL, fields and records. Each partner/group is given a specific amount of time to research their assigned concept and think of a unique way to 'teach it' to the rest of the class. At an identified time, invite each pair to 'teach' what they have learnt to the class. In turn, the class can then review/evaluate their thoughts on what they have been taught.

Answers

Activity 4.1

Variable	Data type
firstName	String
lastName	String
Initial	String
Age	integer

Activity 4.2

```
myString = input("Please enter a string of letters")
myChar = input("Please enter the character you wish to search for")
times = 0
for index = 0 to mystring.length - 1
    if myString(index) == myChar  then
        times = times + 1
    endif
next index
print (times)
```

© Cambridge University Press

Activity 4.3

```
myString = File(RevisionNotes) – telling the program to search external files
testString = ""
times = 0
for index = 0 to myString.length -8
        testString = teststring + myString(index)
        for test = 1 to 7
                testString = testString + myString(index + test)
        next test
        If testString == "variable" then
        found = "Yes"
        times = times + 1
endif
testString = ""
next index
If found == "Yes" then
print("It was found at index " + position + times + " times")
else
print("Sorry. Not found.")
endif
```

Activity 4.4

```
firstname = input ("please enter your first name")
surname = input ("please enter your surname")
fullname = firstname + " " + surname
print ("Hello" + fullname + " , how are you?")
```

Activity 4.5

```
array cars [5]
for index = 0 to 4
        Response = input("please enter the name of a car" + (index + 1))
        cars[index] = response
Next index
print ("All cars have now been entered")
```


© Cambridge University Press

Activity 4.6

```
for index = 0 to 4
    print (cars[index])
Next index
```

Activity 4.7

```
times = 0
for index = 0 to sales.length -1
    if sales[index] == "99" then
        times = times + 1
    endif
next index
print ("there were " + times + " 99 ice creams sold today")
```

Activity 4.8

This algorithm assumes that the array alphabet [26] has already been set up and populated:

Alphabet array iterate and at each cycle compare with variable computer

```
searchString = "computer"
for stringIndex = 0 to searchString.length - 1
    letter = searchString(stringIndex)
    found = False
    arrayIndex = 0
    while found = False AND arrayIndex < arrayAlphabet.length
        if arrayAlphabet[arrayIndex] == letter then
                found = True
                print(letter + " has the index value of " + arrayIndex)
        else
                arrayIndex = arrayIndex + 1
        endif
    endwhile
next stringIndex
```

© Cambridge University Press

Activity 4.9

This algorithm assumes that the array *marks* has already been populated.

```
index = 0
min = marks[index]
sum = 0
while  index < marks.length
        if marks[index] < min then
                min = marks[index]
        endif
        sum = sum +  marks[index]
        index = index + 1
endwhile
print("Your lowest mark is: " + min)
print( "Your mean mark is: " + sum/marks.length)
```

Activity 4.10

```
array marks =  [6, 9, 2, 5, 8, 3, 9, 9, 10, 9, 5, 7, 10]
array pass =  [13]
index = 0
for numb =  0 to marks.length - 1
        if marks[numb] >= 5  then
                pass[index] = marks[numb]
                index = index + 1
        endif
next numb
```

Activity 4.11

(a) orange, yellow, green

(b) yellow, green, blue

(c) green

Activity 4.12

```
for index = 1 to exam1.length - 1
        if exam1[index] <  50 then
                exam1[index] =  exam1[index] + 5
        elseif exam1[index] > 50 then
                exam1[index] = exam1[index] + 10
        endif
```


© Cambridge University Press

```
next index
```

Activity 4.13

(a) blue

(b) red

Activity 4.14

This algorithm assumes that the two-dimensional array *results* have already been initialised and populated and that a *.length* method is available that determines the number of elements in a list:

```
total = 0
highestScore = 0
number = results.length -1
bestStudent = ""
for index = 1 to number
    total = total + results[index, 1]                  // Keeps a running total of the scores
    if  results[index, 1] > highestScore then          // Compares score with highest score
        highestScore = results[index, 1]               // updates highestScore
        bestStudent = results[index, 0]                // and best student if appropriate
    endif
next index
meanScore=total/number                                 // Calculates mean score
print( "The mean score was " + meanScore)
print( "The highest score was " + highestScore + " achieved by " + bestStudent)
```

Activity 4.15

Students' own answers

© Cambridge University Press

Chapter 5: Searching and sorting algorithms

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain why sorted lists are of more value than unsorted lists
- describe the bubble sort, selection sort and merge sort algorithms
- use these algorithms to sort lists into ascending and descending order
- describe the linear and binary search algorithms
- use these algorithms to search sorted and unsorted lists
- write code for the implementation of these algorithms.

What your students need to know

Students should:

- be able to use pseudocode to create variables and display algorithms
- be able to create and use arrays.

Vocabulary

- Ascending order
- Descending order
- Bubble sort
- Insertion sort
- Merge sort
- Sequential
- Linear sort
- Binary search

Common misconceptions and other issues

Students should be encouraged to work through the stages of each sorting and searching algorithm, showing the results of each stage.

It is worth stressing that when sorting into ascending order using a bubble sort, the highest unsorted value will be in its correct position at the end of each pass.

The efficiency of the different algorithms should be stressed, particularly when comparing linear and binary searches.

Skills and coding

- Maths skills:
  - Median (**Activities 5.7 and 5.8**).
- Coding skills:
  - Use of pseudocode
  - Nested loops
  - Creating and populating arrays
  - Finding the length of an array
  - Using loops to traverse an array
  - Using comparison operators.

© Cambridge University Press

Skills and coding for non-specialist teachers

1 Bubble sort

In a bubble sort, the first two items (i.e. items 1 and 2) are compared and are swapped round if they are not in the required order. Then the next pair (items 2 and 3) are compared. This continues until the end of the list.

If they are being sorted into ascending order, the item with the highest value will be in its correct position at the end of the first pass.

Passes are repeated until there are no swaps.

The code for a bubble sort is given after **Activity 5.2** of the Student Book.

This pseudocode will check through the entire list each time. The students could be asked to write the code for a bubble sort and amend it so that it does not check the numbers at the end of the list, which are already in their correct positions each time it carries out a pass. This reinforces the consideration of algorithm efficiency.

The algorithm can be adapted by decreasing the length of the list before the next iteration.

This line should be added at the end of the algorithm:

```
next x
N = N – 1
endwhile
```

Bubble sort is explained in the Student Book and the solution to **Activity 5.2**.

2 Insertion sort

In an insertion sort each item is examined in turn and moved into its correct position. If it is lower than items to its left then those items have to be moved to the right to accommodate it.

Insertion sort is explained in the Student Book and the solutions to **Activities 5.3 and 5.4**.

Activity 5.4 asks students to create and test a program to carry out an insertion sort.

A possible algorithm in pseudocode is given below.

Pseudocode	Explanation
S = List of items N = length of list	The variable S is assigned to the array. The variable N is set to the length of the array.
for x = 1 to N	This starts a loop that starts at '1' and ends at the value equal to the length of the array.
value = S[x]	The variable 'value' is assigned the value at the index 'x' of the loop. Arrays start at index 0 and so the loop starts with the second item of the array.
position = x	The variable 'position' is assigned the value of 'x'. So for the first turn of the loop, it will be equal to 1.
while position > 0 AND S[position – 1] > value	This starts a while loop that will continue while the position is greater than 0 so that it does not try to go beyond the start of the array when it compares the number with the one to the left, i.e. it would generate an error if it tried to compare it with the number at an index of 0-1! It will also continue while the number stored in the variable 'value' is less than the value stored in the array at the index to the left, i.e. one less than the index holding the number stored in 'value'.

© Cambridge University Press

Pseudocode	Explanation
S[position] = S[position – 1]	If the number to the left is greater, then it is moved to the index of the original number, i.e. it is moved to the right.
position = position - 1	The variable 'position' is now reduced by one so that the search can continue further to the left.
endwhile	The while loop will now run again with the index position set one place to the left, i.e. one place lower.
S[position] = value	Once the conditions set in the while loop are not met i.e. the start of the array has been reached or the number in the array at the lower index to the left is actually less than the number stored in 'value', then it is written into the array at this index position.
next x	This ends the 'for' loop and it will run again until it reaches the end of the array.

3 Merge sort

Merge sort employs an algorithm based on recursion. It is said to be **divide-and-conquer**, as it breaks the problem into sub problems that are similar to the original problem, recursively solves the sub problems, and finally combines the solutions to the sub problems to solve the original problem. Because divide-and-conquer solves sub problems recursively, each sub problem must be smaller than the original problem, and there must be a base case for sub problems. It has three parts:

Divide the problem into a number of sub problems that are smaller instances of the same problem.

Conquer the sub problems by solving them recursively.

Combine the solutions to the sub problems into the solution for the original problem.

Merge sort is explained in the Student Book and the solution to **Activity 5.5**.

4 Sorting algorithms

Linear and binary search algorithms are very straightforward and students should have encountered them in their daily lives.

Comparisons of the best and worst case scenarios provide a good example of discussing algorithm efficiency.

Sorting algorithms are explained in the Student Book and the solutions to **Activities 5.6 to 5.9**.

The students are not asked to code a binary algorithm but it could be done as an extension activity.

An algorithm, in pseudocode, is shown below.

Pseudocode	Explanation
target = int(input("Please enter a target: "))	A variable to store the item to be searched for is declared as 'target'. In this instance a number is the expected input.
start = 0	The variable 'start' is set to the index number of the first item
end = list.length-1	The variable 'end' is set to the index of the last item of the list.
found = False	The Boolean variable 'found' is set to False. This is used to indicate that the search item has not been found.
while start <= end AND found == False	A 'while' loop is set up.
middle = ((start + end)/2)	The median item is found.
if list[middle] == target then print(target + " is in the list") found = True	If the median number is the target, then 'found' is set to True and the user is informed.

© Cambridge University Press

Pseudocode	Explanation
elseif target < list[middle] end = middle - 1	If the search item is less than the mean, then the variable 'end' is set to the item to the left of the mean – the next item less than the mean.
else start = middle + 1	If the search item is greater than the mean, then the variable 'start' is set to the item to the right of the mean – the next item greater than the mean.
endif	
endwhile	

If found = False then
print(target + " is not in the list.")
endif
	The user is informed if the search item has not been found.

Prompting questions

- Why is it important to sort things into an order?
- How many examples can you think of where data has been sorted?
- Bubble sort, insertion sort and merge sort are all different ways to sort data. How do you think these work?

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- Provide students with numbered cards in random order. Tell them to place the cards next to each other and then sort them into order without telling them how. Then ask students how they sorted the data and to write their strategy down. How efficient was their strategy? Could they have done it better? Ask pairs of students to compare their techniques with each other; which one was faster and why? This can then be used to compare against the sorting techniques discussed in the chapter. It works well either as a starter or a plenary.
- Students play a game. With a deck of sorted cards, a player chooses a number secretly. The 'magician' has to work it out using only questions such as 'is it higher or lower?' Another option is to play 20 questions with the class. The ideal questions are those which automatically eliminate at least half the options; such as 'is it male or female?' The popular CS4FN / Teaching London Computing initiative, funded by the Mayor of London, has a detailed outline of activities to teach searching algorithms. **www.cambridge.org/links/kotd4006**

Enrichment activities

- Ask students to investigate the different search and sort algorithms. Can they find the most effective YouTube video that explains the different algorithms and their differences and characteristics?
- Search through **www.cambridge.org/links/kotd4007** and read through the resources. Using inspiration from that style, write an article to explain the different sort algorithms.

Assessment ideas

- Students carry out an investigation to code the different sort algorithms and run them with the same set of data. Can students discover which is the most efficient? Ask them to consider their own criteria for comparison and present their findings and justifications at the end.

© Cambridge University Press

Answers

Activity 5.1

Students' own answers

Activity 5.2

20	15	3	13	9	2	6
15	3	13	9	2	6	20
3	13	9	2	6	15	20
3	9	2	6	13	15	20
3	2	6	9	13	15	20
2	3	6	9	13	15	20
2	3	6	9	13	15	20

Activity 5.3

20	15	3	13	9	2	6
15	20	3	13	9	2	6
3	15	20	13	9	2	6
3	13	15	20	9	2	6
3	9	13	15	20	2	6
2	3	9	13	15	20	6
2	3	6	9	13	15	20

Activity 5.4

This algorithm assumes that the array *testResults* has already been initialised and populated and that a *.length* method is available that determines the number of elements in a list. Please see algorithm on page 33.

```
number = testResults.length
for index = 1 to number - 1:
        currentMark = testResults[index]
        position = index
        while position > 0 AND testResults[position - 1] > currentMark
                testResults[position] = testResults[position - 1]
                position = position - 1
        endwhile
        testResults[position] = currentMark
next index
print(testResults)
```


© Cambridge University Press

Activity 5.5

20	15	3	13	9	2	6

20/15/3/13. 9/2/6

20/15. 3/13. 9/2. 6

20. 15. 3. 13. 9. 2. 6

15/20. 3/13. 2/9. 6

3/13/15/20. 2/6/9

2/3/6/9/13/15/20

Activity 5.6

This algorithm assumes that the array *popularNames* has already been initialised and populated with the hundred most popular names.

```
found = false
index = 0
name = input('Please enter the name you want to search for: ')
while found == false AND index < 100
        if name == popularNames[index] then
                found = true
        endif
        index = index + 1
endwhile
if found == true then
        print(name, 'is in the list.')
else
        print(name, 'is not in the list.')
endif
```

Activity 5.7

3	15	21	27	33	39	42	48	56	60	66	67	69
3	15	21	27	33	39							
3	15	21										
21												

Activity 5.8

- Pick median 15
- Too low, so select right hand side, 6 numbers left so choose 56
- Too high, so answer is 45

© Cambridge University Press

Activity 5.9

- Find the length of the array
- Start (of search items) equals 0
- End of search items equals length of array – 1
- While Start is less than or equal to End
- Middle equals (start + end) / 2
- If Middle is equal to number entered tell the user and stop the loop
- If middle is less than number entered then Start equals Middle + 1
- If middle is greater than number entered then End equals Middle – 1
- End of while loop
- Inform the user that the number is not present

© Cambridge University Press

Chapter 6: Input and output

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain why user input is needed
- describe ways in which data input can be validated
- format outputs
- work with text files.

What your students need to know

Students should:

- be able to use pseudocode to create variables and display algorithms
- be able to use selection and definite and indefinite iteration
- be able to create and use one and two-dimensional arrays.

Vocabulary

- Logical error
- Syntax error
- Validation
- Presence check
- Range check
- Length check
- Transcription error
- Modulus check digit
- Authentication
- File handle
- Write mode
- Read mode

Common misconceptions and other issues

It should be stressed that valid data is not necessarily correct. For example, when the students' details are being entered onto the school system, a year group of 10 would be a valid entry but the student in question might not be in year 10. The entry would be valid but incorrect.

When checking that data has been entered by a user, a variable initialised as a blank string can be used.

Students should be encouraged to investigate the on-screen formatting commands in the language they are studying.

In the Student Book, the OCR pseudocode commands are used when working with text files.

In the pseudocode only static arrays are implemented but when reading data items from a text file, dynamic arrays and the 'append' command are far more user-friendly as the number of data items does not have to be known in advance.

Students should be encouraged to investigate commands available in the language they are studying.

© Cambridge University Press

Skills and coding

- Coding skills:
  - Use of pseudocode
  - Nested loops
  - Creating and populating arrays
  - Finding the length of an array
  - Using loops to traverse an array
  - Opening and closing text files
  - Writing to and reading from text files.

Skills and coding for non-specialist teachers

1 Text files

When using text files, file handles are used. This is a reference to the file used by the operating system when writing to or reading from the file.

```
myFile = openWrite("samplefile.txt")
```

or

```
myFile = openRead("samplefile.txt")
```

In both examples above, a variable or file handle is given to the file.

The file should be closed when file operations are complete.

```
myFile.close()
```

When a file is opened in write mode, a new file will be created or an existing file with that name will be overwritten.

Most programming languages allow files to be opened in append mode so that extra data can be written to them.

This is not available in the OCR pseudocode and students should be encouraged to investigate extra commands in the language they are studying.

The use of text files is explained in the Student Book and the solution to **Activity 6.5**.

Prompting questions

- Can you think of one example where GIGO is especially important?
- What is the difference between validation and verification?
- Can you think of any examples of validation routines that you have encountered? What data were you entering, what do you think the validation rule was?
- Why are check digits important?
- What is authentication?
  - Why is this important?
  - What are the different ways we can authenticate a user identity?
- What is identity theft?

© Cambridge University Press

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- List all the devices that you can think of, then categorise them. Are they primarily input, output or something else?
- Give students a program with errors in it. You might even ask them to examine a program they have written that possibly still doesn't work. In pairs, ask them to circle/highlight and label all the syntax and logical errors, clearly stating which error is which and why.

Enrichment activities

- Examine the last program/algorithm that you wrote. Have you included any validation routines? Where and how could validation routines be included? Modify your program/algorithm to include validation.
- Investigate the different algorithms and code samples available online to help calculate check digits. Use this to write your own program to calculate check digits. Can you extend the program to work for a 10 digit ISBN number?
- Investigate the different authentication techniques carried out by social networking sites, banks, online ordering sites, etc. What are the different methods used and why do they work?
- Can you find a recent example of identity theft, or a security breach that might result in identity theft? How did this breach/theft occur and how could it have been prevented?

Assessment ideas

- Ask students to write a program that will enable a record of information to be constructed. This record could be about anything, including: friends' personal and birthday information, music collection, game sales, test and assessment results, etc. The important thing is that a variety of data is possible for input. The program should attempt to validate each field or item of information entered. Check that students have constructed appropriate validation routines.

Answers

Activity 6.1

```
Age = input("please enter your age")
if Age >= 17 then
        print ("You can apply for a driving licence")
else
        print ("you are too young to apply for a driving licence")
endif
```

© Cambridge University Press

Activity 6.2

While this activity is best carried out by writing the program for it, the logic for the rule would be: Valid Password: Password_Entered.length >= 9

```
password = input("Please enter your password.")
if password.length >= 9 then
        print("The length of your password is OK.")
else
        print( "Your password must have at least nine characters. Yours has only " + password.length)
endif
```

Activity 6.3

```
checksum = 0
number = input("Please enter a 7 digit number.")
while number.length != 7
        print("Number not valid.")
        number = input("Please enter a 7 digit number.")
endwhile

d1 =int(number(0))*8
d2 =int(number(1))*7
d3 =int(number(2))*6
d4 =int(number(3))*5
d5 =int(number(4))*4
d6 =int(number(5))*3
d7 =int(number(6))*2

sum=(d1+d2+d3+d4+d5+d6+d7)
mod=sumMOD 11
d8 = 11 - mod
if d8==10 then
        d8='X'
endif
finalNumber = str(number) + str(d8)
print("Your 8 digit Number is: " + FinalNumber)
```


© Cambridge University Press

Activity 6.4

The activity asks the students to adapt the existing algorithm.

```
userEntry = ""
foundName = 0
passwordFound = False
while userEntry == "" AND passwordFound == False
        userEntry = input("Please enter your user name.")
        usersLen = users.length
        for index = 0 to usersLen - 1
                if userEntry == users[index, 0] then
                        foundName = 1
                        attempts = 0
                        while attempts <3 AND passwordFound == False:
                                passwordEntry = input("Please enter your password")
                        if passwordEntry == users[index, 1] then
                                                print("User name and password are correct.")
                                                passwordFound = True
                                else:
                                                print("Sorry the password is incorrect.")
                                                attempts = attempts + 1
                                endif
          ENDWHILE
                        if attempts >= 3:
                                        print("Account has been locked!")
                        endif
                        endif
        next index
        if foundName == 0:
        print("User name is not recognised.")
                userEntry = ""
        endif
endwhile
```

© Cambridge University Press

Activity 6.5

```
//writing the high scores from the array into the file
myFile = openWrite("HighScores.txt")
for index = 0 to 4
        myFile.writeLine(scores[index])
next index
myFile.close()
//writing the scores from the file back into the array
myFile = openRead("HighScores.txt")
for index = 0 to 4
        scores[index] = myFile.readLine()
next index
myFile.close()
```


© Cambridge University Press

Chapter 7: Problem solving

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain what is meant by computational thinking
- explain what is meant by *decomposition* and *abstraction* and use them to solve problems
- create algorithms to solve problems that you have analysed
- explain what is meant by top-down and bottom-up problem solving
- create structured programs using procedures
- follow the systems development cycle to analyse problems, design and implement solutions and test the outcomes.

What your students need to know

Students should:

- be able to use pseudocode to create variables and display algorithms
- be able to use selection and definite and indefinite iteration
- be able to ask for and incorporate user input
- be able to use trace tables.

Vocabulary

- Computational thinking
- Decomposition
- Abstraction
- Pattern recognition
- Top-down problem solving
- Bottom-up problem solving
- Structured programming
- Modules
- Subroutine
- Function
- Procedure
- Call a subroutine
- Argument
- Parameter
- Global variable
- Local variable
- Systems development cycle
- Identification and analysis
- Logical errors
- Implementation
- Syntax error
- Integrated development environment
- Source code editor
- Alpha testing and test plan
- Valid test

© Cambridge University Press

- Boundary test
- Erroneous test
- Evaluation
- Maintenance

Common misconceptions and other issues

Abstraction can be thought of as removing unnecessary details to get to the heart or essence of something.

It can be introduced by considering abstraction in everyday situations such as:

- Creating mental models of objects such as cars, houses, animals, etc. so that we can communicate with each other.
- Levels of abstraction can be illustrated by our use of machinery without knowing exactly how it works, for example, starting and driving a car without knowing how the combustion engine works.
- In a similar way, when we use the print() function, we do not need to know all of the coding involved in making this happen.
- When we use a high level language, we do not need to know the actual machine code as it is translated for us by a compiler or interpreter. We are working at a higher level of abstraction. Assembly language is a low level of abstraction as it is more similar to machine code.

A subroutine is a set of instructions designed to perform a frequently used operation in a program.

It is 'called' by the main program.

- A function returns a value back to the main program.
- A procedure does not return any data to the main program.

When a subroutine is called, the data it needs (the parameters) are passed to it as arguments from the main program.

- The data is passed as an 'argument' and accepted as a 'parameter'.

It is also worth pointing out that a function can be called from within another function.

The first function can pass arguments to the second one that can return values to the first one.

Skills and coding

- Coding skills:
  - Creating functions with parameters
  - Calling functions with arguments.

Skills and coding for non-specialist teachers

1 Functions

When a function is called the data it needs to process are passed to it as arguments.

In the Student Book, **Activity 7.1** is an exercise on decomposition and abstraction involving the calculation of the approximate cost of a car journey.

© Cambridge University Press

This could be coded using a function:

Pseudocode	Explanation
function cost (distance, mpg, petrol_price)	The function is defined with the identifier 'cost' with the parameters 'distance', 'mpg' and 'petrol_price'. These are local variables used only within the function.
cost = (distance/mpg) * petrol_price	This statement will calculate the cost of the journey and store the value in the variable 'cost'.
return cost	This statement returns the value of 'cost' to the statement in the main program that is called the function.
endfunction	This denotes the end of the function definition.
journey_distance = input("Please enter journey distance in miles")	These statements ask for user input. The values are stored in the global variables, journey_distance, miles_per_ gallon and price_per_gallon.
miles_per_gallon = input("Please enter the average miles per gallon for the car")	
price_per_gallon = input("Please enter the price of one gallon of petrol")	
journey_cost = cost(journey_distance, miles_per_gallon, price_per_gallon)	This statement calls the function 'cost'. The values of the three variables are passed to it as arguments. They are passed in the same order as the three parameters listed in the function. The function will return its result to the variable journey_ cost.
print("The approximate cost of the journey will be " + journey_cost)	The journey cost is output for the user.

The local variables used to store the same values as the global variables are given different identifiers.

The values given as arguments from the main program must be in the same order as expected by the parameters in the function.

Prompting questions

- Can you think of any examples in your everyday life that illustrate or use computational thinking?
- Decomposition helps us to solve problems and make them more manageable by breaking them down into smaller parts. Look around you at the things you do every day; what examples can you see that use decomposition?
- "A game such as Sim City is an example of abstraction" Do you agree or disagree with this statement? Why?
- What is structured programming?
- The opposite of the 'top-down' approach is known as the 'bottom-up' approach. What do you think this means? How would this approach work?
- What is beta testing?
- When developers release software (usually for free) in Beta version, what does this usually mean? Why are they doing it?

© Cambridge University Press

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- Carefully consider the activities that you do regularly (these could be anything from jigsaw puzzles to D&T projects or more). Pick one that you think uses computational thinking. Break this down and describe the activity; which strands of computational thinking it covers, how and why.
- Give students an example of some code and ask them to highlight examples of the following (possibly in different colours, or labelled and annotated):
  - local variables
  - global variables
  - functions and/or procedures
  - arguments and parameters
  - iteration
  - selection statements
  - array.
- Give students a small program to test. Ask them to design and carry out a test plan to see if it works.
- Choose one of the Computational Word Games from the Playful Computing page on the Digital Schoolhouse website (**www.cambridge.org/links/kotd4008**). Select the stack of words under the programming and random categories to test students' knowledge and understanding of some of the key words. You can easily add your own words to this stack. As an interesting variation ask students: how could the rules of the game be adapted and extended?

Enrichment activities

- Investigate the top-down and bottom-up approaches to computing. These are often different schools of thought to solving a problem that have been used in many areas of computing. Focusing on a specific area, can you find examples of how the approaches have been used in computer science research and development? For example, a top-down approach in robotics generally implies that the researchers have focused on the higher order things first, such as talking and activities closer to human level. The bottom-up approach instead focuses on a single 'sense'. Which robots have been developed as a result of the two approaches?
- The Playful Computing section of the Digital Schoolhouse website (**www.cambridge.org/links/kotd4009**) uses unplugged activities to teach computational thinking. Ask students to select one of the activities and investigate it as a group. When reporting back to the class, they should attempt to deliver the activity and explain how and why it maps to computational thinking.
- Investigate and search for some software that is being released in Beta version. Find out what the developers are offering and what they expect in return. Write a brief summary and exchange notes with peers in the class.

Assessment ideas

- Ask students to complete the final challenge for the chapter. Encourage them to follow good practice and guidance when documenting their solution. Students should submit a full testing plan with evidence of testing carried out on their work when they submit their work.

© Cambridge University Press

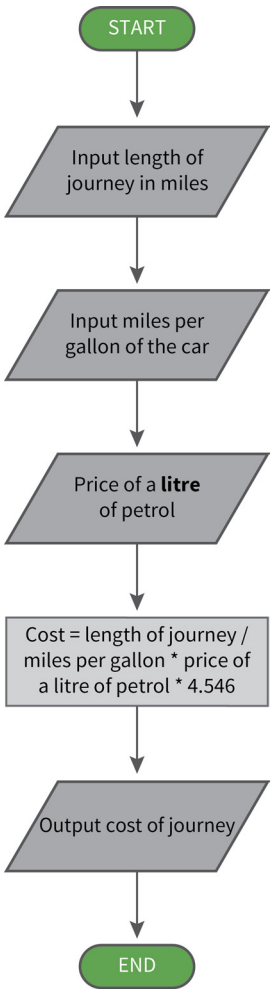
Answers

Activity 7.1

Possible sub tasks might include amongst many others:

- Calculate the length of the journey in miles (or km)
- Establish the cost of petrol
- Find out how many miles (or km) can be driven per litre of petrol
- Calculate the cost of the journey.

Activity 7.2



© Cambridge University Press

Activity 7.3

This algorithm assumes that a *random* function is available that generates a random number in the range x to y.

```
function diceThrow():                          // simulates a dice throw
    throw = random(1, 6)
    return(throw)
endfunction
//start of main program
highestScore = 0                               // keeps track of highest score
anotherGo = input('Play the game (y/n)? ')
while anotherGo == 'y' OR anotherGo = 'Y'      // allows for upper and lower case entry
    total = 0
    total = total + diceThrow() + diceThrow() + diceThrow()
        print('The total this time is:', total)
    if total > highestScore then
        highestScore = total
    endif
    anotherGo = input('Play the game again (y/n)? ')
endwhile
print('The highest score you achieved was: ', highestScore)
```

Activity 7.4

```
function message(one, two)
    print("Hello " + two + " " + one)
endfunction
firstName = input("Please enter your first name.")
secondName = input("Please enter your surname.")
message(firstName, secondName)
```

Activity 7.5

Input 1	Input 2	Solution
3	6	2
4	7	2.5
5	8	3

© Cambridge University Press

Activity 7.6

```
function dogAge()                                    // calculates human equivalent age of a dog
    dogYears = int(input('Enter the age of your dog: '))   //typecasts dogYears as integer
    if dogYears == 1 then
        humanEquivalent = 12
    elseif dogYears == 2 then
        humanEquivalent = 24
    else
        humanEquivalent = 24 + (dogYears - 2) * 4
    endif
    return humanEquivalent
endfunction
function catAge()                                    // calculates human equivalent age of a cat
    catYears = int(input('Enter the age of your cat: '))   //typecasts catYears as integer
    if catYears == 1 then
        humanEquivalent = 15
    elseif catYears == 2 then
                    humanEquivalent = 24
    else
        humanEquivalent = 24 + (catYears - 2) * 4
    endif
    return humanEquivalent
endfunction
//start of main program
anotherGo = 'y'
while anotherGo == 'y' OR anotherGo == 'Y'           // allows for upper and lower case entry
    pet = input('1. Cat, 2. Dog ')
    if pet == '1' then
        print('The human equivalent age of your pet is ', catAge())
    elseif pet == '2' then
        print('The human equivalent age of your pet is ', dogAge())
    else
        print('Invalid choice')
    endif
    anotherGo = input('Do you want to use the calculator again (y/n)?')
endwhile
```

© Cambridge University Press

Chapter 8: Binary and hexadecimal

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain how data is represented by computer systems
- explain why the binary system is essential for computer processing
- convert binary numbers into denary and vice versa
- carry out addition, subtraction, multiplication and division on binary numbers
- use left and right shifts when multiplying or dividing binary numbers by powers of 2
- explain why hexadecimal numbers are used
- convert between binary, denary and hexadecimal.

What your students need to know

Students should:

- have basic maths skills
- be able to use pseudocode or a programming language to create a program to convert between denary, binary and hexadecimal for the final challenge in the chapter.

Vocabulary

- Binary
- Binary digit
- Number bases
- Denary
- Place values
- Overflow error
- Binary shifts
- Hexadecimal

Common misconceptions and other issues

The value of any digit in any number system is dependent on its place value.

In binary, place values increase in powers of 2, in denary, values increase in powers of 10 and in hexadecimal in powers of 16.

Students often have trouble grasping that 0 is a digit or number and in denary there are 10 digits, 0 to 9, and in hexadecimal there are 16 digits, 0 to 15.

When converting between kilobyte, megabyte, gigabyte, etc. the specification does not state whether the binary or denary prefix should be used.

The binary prefix multiplies a value by powers of 2 whilst the denary prefix by powers of 10.

Unit	Denary prefix		Binary prefix	
Kilobyte	10^3 bytes	1000 bytes	2^{10} bytes	1024 bytes
Megabyte	10^6 bytes	1000 kilobytes	2^{20} bytes	1024 kilobytes
Gigabyte	10^9 bytes	1000 megabytes	2^{30} bytes	1024 megabytes
Terabyte	10^{12} bytes	1000 gigabytes	2^{40} bytes	1024 gigabytes

In the Student Book, the denary prefix is used.

© Cambridge University Press

Skills and coding

- Maths skills:
  - Place values in binary, denary and hexadecimal
  - Converting 8 bit binary numbers to denary
  - Converting denary numbers up to 255 to binary
  - Binary addition
  - Binary shifts for multiplication and addition
  - Converting between hexadecimal, binary and denary.
- Coding skills:
  - Use of pseudocode or a programming language to create a program to convert between denary, binary and hexadecimal for the final challenge in the chapter.

Skills and coding for non-specialist teachers

The final challenge allows students to create a structured program using procedures.

- A menu can be used for users to select the type of entry required, i.e. binary, denary or hexadecimal.
- Validation can be used, for example, to ensure that 8 digits are entered for a binary number and they are either 0 or 1. A denary number should be equal to or less than 255 and a hexadecimal one should be less than or equal to FF.
- Procedures can be called to carry out the conversions into the other number systems.

Prompting questions

- Why do you think binary consists of only two states (1 and 0)?
- When you purchase a device such as a smartphone, tablet or games console, etc. it is usually only available in very specific memory capacities, such as: 16 gb/32 gb/64 gb/128 gb, etc. Why is it always these numbers that are used? What is special about them and how does this relate to binary numbers?
- If computers can only work in binary data, why do we need to devote so much time to converting it to decimal and hex? Why not just work in binary?
- Why and when would developers prefer to use the hex number system rather than binary?
- Where are hex values most commonly seen? Why are they better at representing this data than binary? (*Answer: most commonly used to represent true colour and used as colour references, i.e. 000000 is white and FFFFFF is black*)

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- An excellent starter to allow students to understand counting in binary is to give them a set of cards and allow them to count the dots. Resources and a lesson plan for the activity can be found at CS Unplugged (**www.cambridge.org/links/kotd4010**).
- Also by CS Unplugged is a Card Flip magic trick that actually demonstrates how computers detect errors using parity bits. It's excellent at showing how error detection works, but also allows students to see how a combination of bits that can only ever be in two states can actually store a lot of data. Resources for the activity can be downloaded here: **www.cambridge.org/links/kotd4011**
- Carry out a web search for animations explaining binary and hexadecimal number systems. Which one do you think is the best one and why?

© Cambridge University Press

Enrichment activities

- All number systems usually follow specific rules. For example, they all have a base value (in binary it's 2, in hexadecimal its 16 and in denary its 10); they all use positional notation and the number of characters available within that number system is directly linked to its base number. With those factors in mind, it is possible to create your own representation system using any number as a base, for example, 'Septimal' that could be a base 7 number system. Or alternatively, you don't have to use numerical characters at all, you could devise your own characters completely. Ask students to devise their own data representation / number system and then present it to the class.
- As an extension of the above activity, can they work out how you could convert data from an existing number system, such as decimal or binary, into their own system? Perhaps they could set their peers some exercises to complete.

Assessment ideas

- Ask students to spend some time looking at the structure of exam questions and mark schemes. Then ask them to devise their own set of exam questions based around what they have learnt in this chapter. For each question, they should also devise a mark scheme. Then swap papers and ask students to answer questions written by one of their peers. The marking and mark schemes should also be subsequently distributed amongst the class for the peer marking exercise. Groups of students should then come together to discuss the quality of the questions, the accuracy of the mark schemes and judgements based on correct/incorrect answers.
  - As an extension to this, students can rate the quality of their peers' questions and mark schemes and these can be used as part of the assessment data gathered from students.
  - Note: this exercise can be repeated for most if not all topics on the course.

Answers

Activity 8.1

- 0000
- 0001
- 0010
- 0011
- 0100
- 0101
- 0110
- 0111
- 1000
- 1001
- 1010
- 1011
- 1100
- 1101
- 1110
- 1111

© Cambridge University Press

Activity 8.2

- 205
- 68
- 170
- 240
- 188

Activity 8.3

- 00001101
- 01000101
- 10000011
- 11000111
- 11110101

Activity 8.4

Observe students' attempts at binary counting with fingers.

Activity 8.5

- A bit is a single value, a nibble is 4 bits, a byte is 8. There are 1024 bytes in a kilobyte and 1024 kilobytes in a megabyte and so on. Therefore, a gigabyte is 1024 megabytes or 1,073,741,824 (2^{30}) bytes specifically.
- 2.048GB
- 1e+9 or $1^{*}10^{9}$

Activity 8.6

- 10001100
- 10100110
- 11000010

Activity 8.7

- 01101000
- 11010000
- 11011100

Activity 8.8

- 00000101
- 00001000
- 00101101

© Cambridge University Press

Activity 8.9

Students' own answers

Activity 8.10

- Decimal: 196; Binary: 11000100
- Decimal: 70; Binary: 01000110
- Decimal: 250; Binary: 11111010
- 60
- C9
- 8D

© Cambridge University Press

Chapter 9: Binary representations

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain how characters are represented in binary
- calculate the ASCII code for any character
- calculate the size of a text file
- explain how images are represented in binary
- calculate the size of an image file
- explain how sound is represented in binary
- calculate the size of an audio file
- explain the disadvantages of large image and audio files
- explain how file compression reduces the size of files
- explain the differences between lossless and lossy file compression.

What your students need to know

Students should:

- know why the binary number system is used for the operation of computers
- be confident in using binary numbers and converting them to hexadecimal and denary
- be confident in using the terms bit, byte, kilobyte and megabyte and be able to convert between them
- be able to understand and create algorithms using the OCR pseudocode
- be able to construct algorithms using sequence, selection and iteration
- be able to apply string manipulation techniques.

Vocabulary

- Character set
- ASCII code
- Unicode
- Pixel
- Resolution
- Colour depth
- Metadata
- Analogue and digital
- Sampling
- Sample rate
- Bit depth
- Compression
- Redundancy
- Lossless compression
- Lossy compression
- Run-length encoding

© Cambridge University Press

Common misconceptions and other issues

When discussing digital images, there is often confusion around the terms 'image size' and 'resolution'.

The size of an image is determined by the number of pixels and the dimensions are given as width and then height, for example, 640 × 480 would mean 640 pixels in width and 480 pixels in height, so giving a total of 307 200 pixels.

Resolution is expressed in pixels per inch, or ppi, and is therefore influenced by the size of the displayed image. Two images having the same image size would have different resolutions if they were displayed at different numbers of pixels per inch.

Skills and coding

- Maths skills:
  - Converting between binary, hexadecimal and denary
  - Converting between bit, byte and megabyte
  - Calculating file sizes of digital images (W × H × D) and digital sound files (sample rate × bit depth × number of channels × length (in seconds)).
- Coding skills:
  - Traversing a string
  - Returning an ASCII code for a character in a string
  - Inserting a character using its ASCII code
  - Using subroutines, selection and iteration when creating an algorithm and then coding the program to compress an image file.

Skills and coding for non-specialist teachers

The OCR pseudocode now has functions to return the ASCII code for a character and return the character for a code. These are:

ASC() and CHR()

Therefore ASC('D') would return the number 68 and CHR(67) would return the letter 'C'.

These functions are common in programming languages.

In Python, the functions are:

ord() to return the denary code from a character

chr() to return the denary code for a character.

Ord('a') will return 97.

chr(97) will return 'a'.

The final challenge asks the students to code an algorithm to carry out run-length encoding. This task will consolidate the learning from previous chapters including iteration, selection and the use of two-dimensional arrays.

© Cambridge University Press

A possible solution, in pseudocode and the Python programming language, is given below:

Pseudocode	Explanation
text = input("Please enter the text: ")	The user is asked to enter the string to be encoded. It is stored in the variable 'text'.
runText = ""	This variable is declared to hold the 'runs' when the string is evaluated.
run = 0	This variable will store the length of each run.
code = ""	The variable 'code' is given the value of an empty string.
length = text.length	The length of the string is stored in the variable 'length'.
if length == 0 then runText = ""	This checks that some text has been entered. If not, the 'run' is a blank string.
elseif length == 1 then	This checks if the there is only one character in the string.
runText = text	If so, the 'run' is just that character and the length of the run is 1.
run = 1	
Else	If the length of the string is greater than 1 then the following code is called.
index = 0	The variable 'index' is set to 0 for the first character in the string.
runText = text(index)	The variable 'runText' is given the value of this character.
run = 1	As there is at least one instance of this character, the variable 'run' is set to 1.
while index < length - 1	A loop is set up to check the rest of the characters, from the character at index 1 to the last index of the string the length of the string minus 1.
if text(index + 1) == runText	If the character at the next index position is the same as the present one, then the variable 'run' is incremented by 1.
run = run + 1	
else	If the next character is different, then the value of the present character and its run length are added to the variable 'code'. The value in the 'run' variable is changed from an integer to a string for this concatenation.
code = code + str(run)	
code = code + runText	
runText = text(index + 1)	
run = 1	
endif	
index = index + 1	The value of the variable 'index' is incremented by 1.
endwhile	The loop is terminated.
endif	
code = code + str(run)	The value of the run is appended to the value of 'code' if it is only 1 character.
code = code + runText	
print(code)	The result of the run-length encoding is printed.

© Cambridge University Press

Pseudocode	Python
text = input("Please enter the text: ")	text = input("Please enter the text: ")
runText = ""	runText = ""
run = 0	run = 0
code = ""	code = ""
length = text.length	length = len(text)
if length == 0 then runText = ""	if length == 0: runText = ""
elseif length == 1 then	elif length == 1:
runText = text	runText = text
run = 1	run = 1
Else	else:
index = 0	index = 0
runText = text(index)	runText = text[index]
run = 1	run = 1
while index < length – 1	while index < length - 1:
if text(index + 1) == runText	if text[index + 1] == runText:
run = run + 1	run = run + 1
else	else:
code = code + str(run)	code = code + str(run)
code = code + runText	code = code + str(run)
runText = text(index + 1)	runText = text[index + 1]
run = 1	run = 1
endif	
index = index + 1	index = index +1
endwhile	
Endif	
code = code + str(run)	code = code + str(run)
code = code + runText	code = code + runText
print(code)	print(code)

Prompting questions

- What is ASCII?
- How does binary affect file size and quality?
- If a single bit is used per pixel to create monochrome graphics, then how are coloured graphics stored?
- What is compression?
- Binary doesn't change, it always consists of 1s and 0s, yet it can represent everything we see on the computer. How does the computer know whether one set of binary is an image or a sound or an application?
- What is metadata and why is this important?

© Cambridge University Press

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- Ask students to discuss in groups how they think computers are able to understand and manipulate text, sound and images considering that they only work in binary. Each group can be assigned to consider either text or sound or images. You might wish to encourage this to be a simple discussion without access to the internet so that students can devise their own rules and estimate how systems work. Their discussions might not be wholly accurate but it is a good starter activity to prep them for the information that they are about to learn.
- What is your favourite colour? Find the hexadecimal code for it and write out the binary equivalent.
- Hex Editor Neo is free software that allows you to view and edit the binary and hex representations of most files. Download it from: **www.cambridge.org/links/kotd4012**. Ask students to import a file of their choice (graphic or sound) to see the binary/hex representation. Can they spot any patterns?
- Investigate the different types of compression?
- What is the binary representation for your name?

Enrichment activities

- Ask students to write messages to each other in binary representation only, it is the job of the recipient to convert the message back to ASCII format and send a reply.
- One useful activity to support data representation is an activity known as 'Paint by Pixels'. Resources for it can be downloaded from: (**www.cambridge.org/links/kotd4013**)
- Ask students to investigate ASCII Art. What is it? How is it related to ASCII and binary representation? Can they collect examples of ASCII Art? Ask students to see if they can develop their own ASCII art work.
- Use Hex Editor Neo to manipulate files in their binary and hex formats. Graphics will work best, by changing the binary or hex colour codes and saving the file. The students will be able to view their altered images.
- JPEG and MP3 files are compressed files. What is the original file format and how are these compressed?

Assessment ideas

- Ask students to write a program that will ask the user to input text and convert it to its binary representation. Convert the returned binary string into denary values.
- Using a program such as Audacity, allow students to manipulate sound files, apply compression techniques and explore the impact on file size and quality.

Answers

Activity 9.1

The ASCII code represents characters.

© Cambridge University Press

Activity 9.2

This algorithm assumes that a *chrCode* function is available that returns the ASCII code of a character.

```
sentence = input('Enter the sentence to decode: ')
numbChars = sentence.length
for index = 0 to numbChars - 1
        asciiCode = ASC(sentence(index))
        print(asciiCode)
next index
```

Activity 9.3

```
sentence = input('Enter a sentence or phrase: ')
print('The size of this sentence/phrase in bytes is:' + ' ' + sentence.length + '.')
```

Activity 9.4

01111110

01111110

01111110

01100110

01100010

01001000

00011000

00111100

Activity 9.5

(a) 267, 480, 480

(b) 2, 457, 600 (640 * 480 * 8)

Activity 9.6

591840000 (41100 * 24 * 2 * 300)

© Cambridge University Press

Activity 9.7

2w3b3w

3w1b4w

3w1b4w

3w1b4w

3w1b4w

3w1b4w

3w1b4w

2w3b3w

= 6 x 8 bytes, that is, 48 bytes

(Without RLE the character would be 8 x 8 bytes in size, that is, 64 bytes.)

© Cambridge University Press

Chapter 10: Programming languages

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- describe the difference between low and high level languages
- explain the advantages of using high level languages
- explain how program instructions are encoded in low level languages
- explain why high level languages need to be translated
- explain the characteristics and use of:
    - an assembler
    - a compiler
    - an interpreter.

What your students need to know

Students should:

- be competent in the use of pseudocode and a programming language.

Vocabulary

- Applet
- Machine code
- Machine language
- Instruction set
- Opcode
- Operand
- Assembly language
- Mnemonic
- Assembler
- High level language
- Low level language
- Compiler
- Interpreter

Common misconceptions and other issues

The instructions for a microprocessor must be presented in machine code that consists of strings of 1s and 0s, arranged as instructions.

Each type or family of processor has its own instructions, known as its instruction set.

To assist programmers, languages have been developed that use commands more related to human languages. They are at a higher level of abstraction.

Assembly language is at a low level of abstraction and the instructions used have a one-to-one relationship with those of machine code.

High level languages, for example, Python, Java and C, are at a higher level of abstraction.

The code from these languages must be translated for the processor. An assembler is used for assembly language and compilers and interpreters for high level languages.

© Cambridge University Press

Skills and coding

- Coding skills:
  - The final challenge introduces the students to a CPU simulator with a limited instruction set. The students are encouraged to investigate the simulator and code simple programs to sort and multiply numbers.

Skills and coding for non-specialist teachers

The Little Man Computer simulator is at: **www.cambridge.org/links/kotd4014**

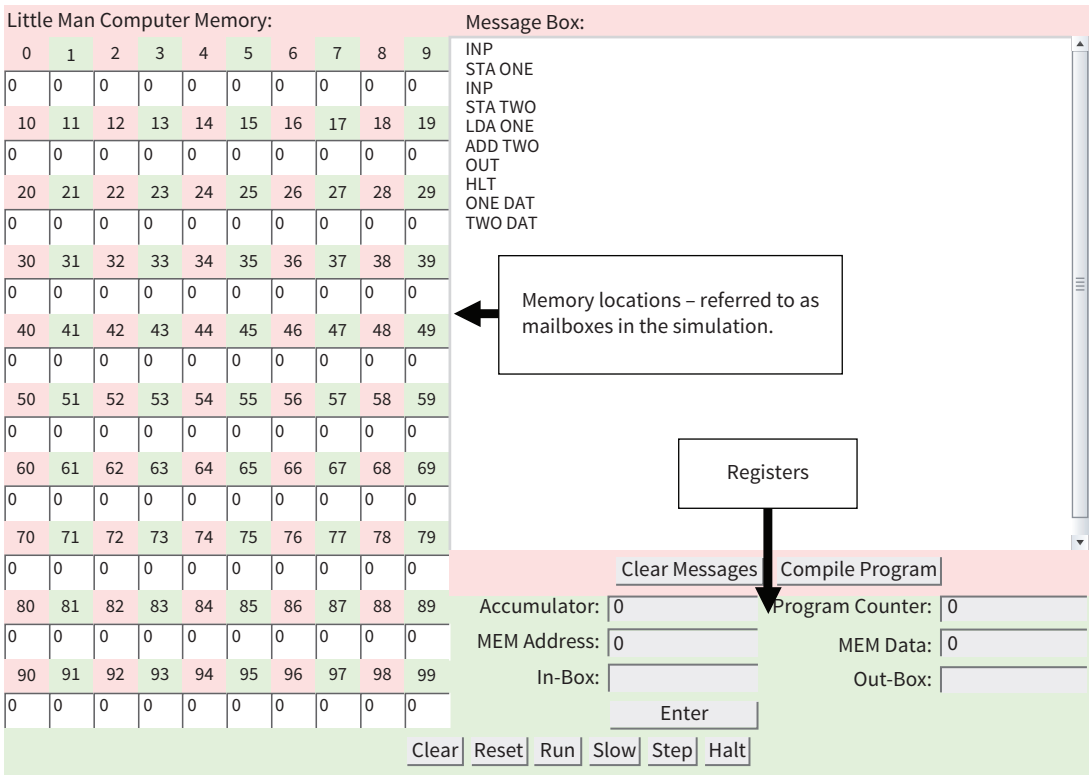
Simulators provide a good introduction to assembly language programming and Little Man Computer, simulator also illustrates the fetch-execute cycle, which is explored in more detail in **Chapter 11**.

The simulator has nine instructions that can be entered using mnemonics.

The following is a simple program to input and store two numbers and then add them together.

INP	Mnemonic for user input for the first number.
STA ONE	Store the first number in memory location labelled ONE.
INP	User input for the second number.
STA TWO	Store the second number in memory location labelled TWO.
LDA ONE	Load the contents of memory location ONE into the accumulator.
ADD TWO	Add the contents of memory location TWO to the accumulator.
OUT	Output the contents of the accumulator.
HLT	Stop execution of the program.
ONE DAT	These commands reserve data locations for the two numbers to be entered.
TWO DAT	

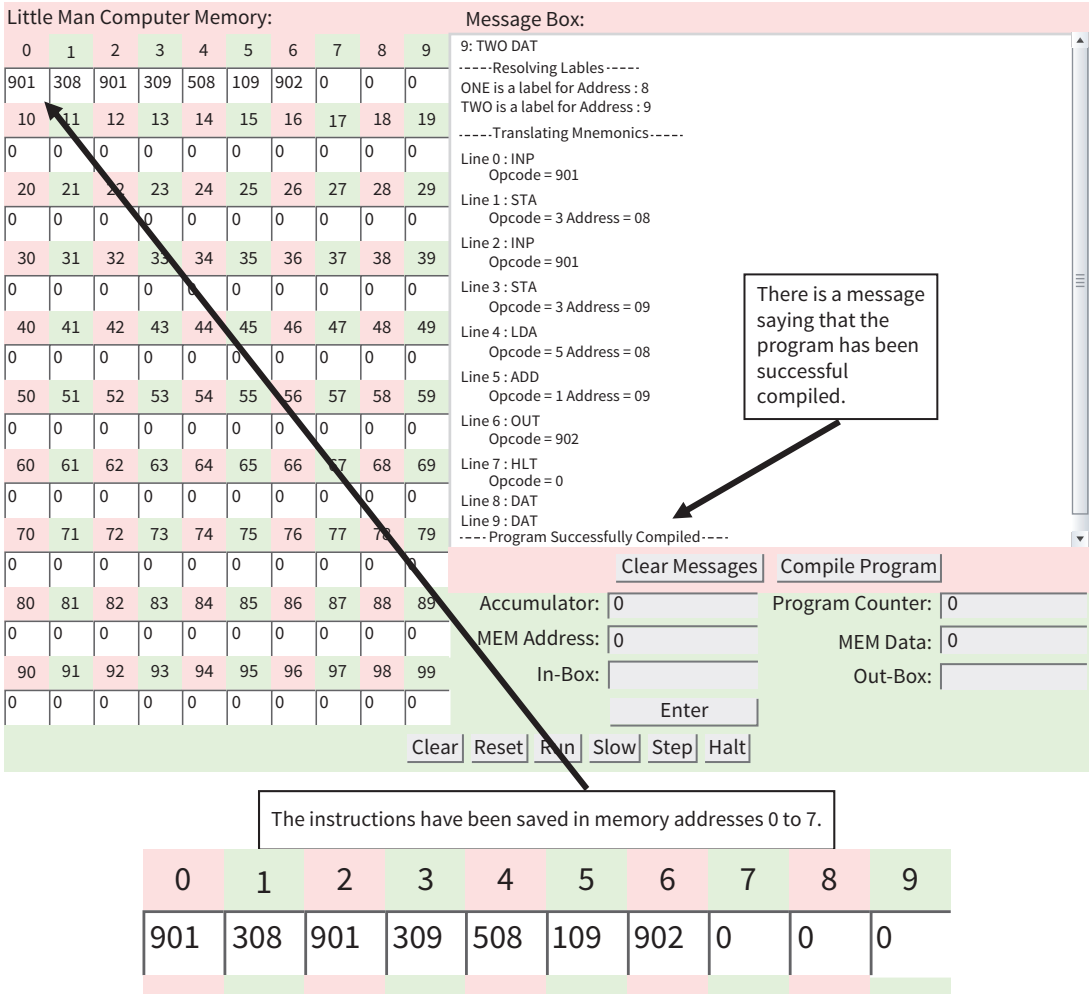
The following diagram shows the program in the message box.



Before the program can be run, it must be compiled into machine code.

This is done by clicking on 'Compile program'.

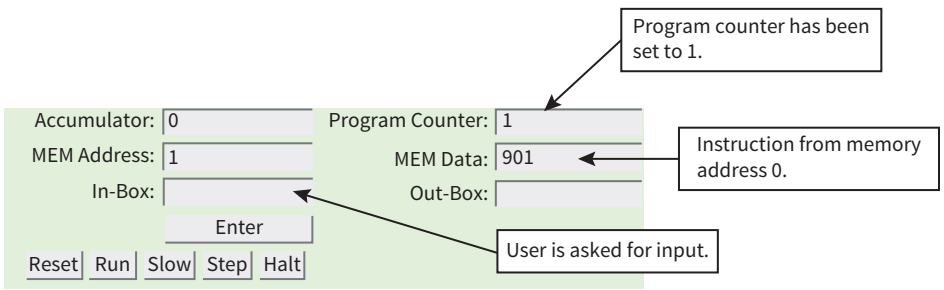
© Cambridge University Press



Address	Opcode	Explanation
0	901	This is the opcode for the mnemonic INP <N> user input.
1	308	3 is the opcode for STA. 08 is the memory location where it will be stored.
2	901	This is the opcode for the mnemonic INP <N> user input.
3	309	3 is the opcode for STA. 09 is the memory location where it will be stored.
4	508	5 is the opcode for LDA and 08 is the location of the data to be loaded into the accumulator.
5	109	1 is the opcode for ADD and 09 is the location of the data to be added to the accumulator.
6	902	902 is the opcode for OUT <N> the contents of the accumulator will be output for the user.
7	0	0 is the opcode for HLT <N> execution of the program will stop.

The program can be run one statement at a time by clicking the 'Step' button.

1



© Cambridge University Press

2

Accumulator: 3 Program Counter: 1
MEM Address: 1 MEM Data: 901
In-Box: 3 Out-Box:
Enter
Reset Run Slow Step Halt

After data has been input and enter button clicked, the number is copied to the accumulator.

3

Accumulator: 3 Program Counter: 2
MEM Address: 8 MEM Data: 3
In-Box: 3 Out-Box:
Enter
Reset Run Slow Step Halt

The second instruction (308) has copied the contents of the accumulator to memory location 8.

0	1	2	3	4	5	6	7	8	9
901	308	901	309	508	109	902	0	3	0

4

Accumulator: 3 Program Counter: 2
MEM Address: 1 MEM Data: 901
In-Box: 6 Out-Box:
Enter
Reset Run Slow Step Halt

The user is asked to enter the second number by the instruction at memory location 2 (901).

5

Accumulator: 6 Program Counter: 4
MEM Address: 9 MEM Data: 6
In-Box: 6 Out-Box:
Enter
Reset Run Slow Step Halt

It is copied to the accumulator and then memory location 9 by the instruction at memory location 3.

0	1	2	3	4	5	6	7	8	9
901	308	901	309	508	109	902	0	3	6

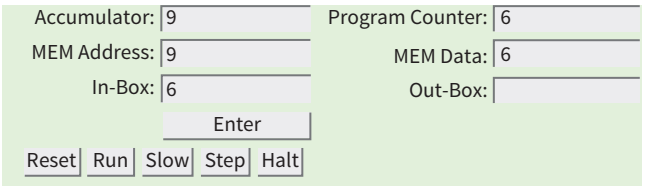
6

Accumulator: 3 Program Counter: 5
MEM Address: 8 MEM Data: 3
In-Box: 6 Out-Box:
Enter
Reset Run Slow Step Halt

The data at memory location 8 is copied to the accumulator by the instruction at memory location 4 (508).

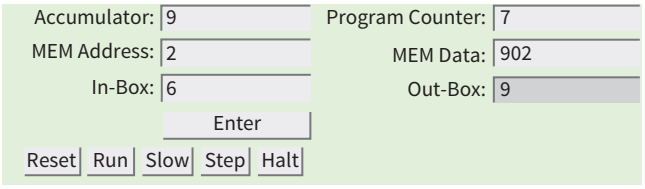
© Cambridge University Press

7



The data at memory location 9 is now added to the accumulator by the instruction at memory location 5 (109).

8



The contents of the accumulator are now copied to the out-box by the instruction at memory location 6 (902). The instruction at location 7 (0) then halts program execution.

Prompting questions

- How many programming languages can you name?
- What is the difference between a high level and a low level language?
- What is an interpreter?
- What is a compiler?
- What is machine code?
- The programming language being learnt by the class, does it use an interpreter or a compiler? Why do you think this is the case?

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- Ask students to list as many programming languages as they know. Are different languages recommended for different purposes?
- Can you find examples of machine code?
- Investigate which programming languages use compilers and which use interpreters

Enrichment activities

- The term 'programming generations' refers to a classification applied to programming languages. Investigate what the different generations are and what they refer to. Can students categorise the different languages into the different categories?
- HTML and PHP are considered to be scripts/scripting languages. Do they use interpreters or compilers? Why do you think this is the case? Justify your answer.
- SQL is often considered to be a fourth generation language. Investigate this to find out what the general consensus is on the issue. What are the most common SQL commands and what is the language commonly used for?
- Prolog is considered to be a fourth generation language. Investigate why this is the case and what the most common commands are. What is the language most commonly used for?

© Cambridge University Press

Assessment ideas

- Write a program that allows the user to enter a series of numbers and return the total value.
  - Write a program in your chosen language to run the above program.
  - Write a program using 'Little Man Computer' to execute the program.
  - Write a machine code to execute the above program.

Answers

Activity 10.1

(a) 0010 0110

(b) 0001 01110001

(c) (i) 0001 1010
0100 10101
0000 11110
(ii) 31

Activity 10.2

Students' own answers

© Cambridge University Press

Chapter 11: Computer systems: hardware

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain what is meant by a computer system
- explain what is meant by an embedded system
- describe the structure of the central processing unit and the functions of its components
- describe the fetch-decode-execute cycle
- explain the need for and role of multiple cores and cache and virtual memory
- describe secondary storage media and the advantages and disadvantages of each.

What your students need to know

Students should:

- be competent in the use of pseudocode and a programming language.

Vocabulary

- Microprocessor
- Bus
- Printed circuit board
- Hardware
- Software
- Embedded system
- Central processing unit
- Von Neumann architecture
- Fetch-decode-execute cycle
- Random access memory
- Read-only memory
- Storage location (address)
- Volatile
- BIOS (basic input/output system)
- Control unit
- Arithmetic and logic unit
- Registers
- Control signals
- Multi-core processor
- Parallel processing
- Multitasking
- Cache memory
- Virtual memory
- Secondary storage device
- Magnetic storage
- Optical storage

© Cambridge University Press

- Solid state storage
- Flash memory
- Cloud storage

Common misconceptions and other issues

Multi-core processors do not produce a proportionate increase in the rate at which programs will run on a computer. For example, programs will not run at twice the speed on a dual-core processor as tasks might be sequential and not run in parallel. One task might not be able to start until another has finished.

Students often confuse ROM with secondary storage. An example to explain the difference is to think of ROM as an old vinyl record, that you could play but not change, whereas a cassette tape could be recorded onto and changed many times.

Because they are usually integral to a computer system, students sometimes think that a hard disk drive is 'primary' storage but it is just another example of a secondary storage device.

Skills and coding

No coding skills are needed for this chapter unless students undertake the final challenge.

Skills and coding for non-specialist teachers

The teaching of this chapter requires no special skills or coding.

Prompting questions

- Computer storage drives currently begin with C:\ which refers to the hard drive, and go upwards with other letters such as D:\, E:\, F:\ onwards referring to CD/DVD drives, removable storage devices and network drives. What happened to A:\ and B:\ drives? What were they used with and why are they no longer referred to?
- What is Moore's law?
- What is the difference between RAM and ROM?
- When purchasing a new computer or games console, the technical specifications of the device will often tell you about how much RAM is built in, and that it is better to buy a device with more RAM.
  - Why is RAM so important?
  - What is RAM responsible for?
  - How does it impact computer performance?
- What is the job of a computer processor?
- Ask students to name/list common computer processor names/brands.
- What has changed in technology to allow computers to shrink in size over the years?
- What was the world's first computer?

© Cambridge University Press

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- Setting up a role play to demonstrate the fetch execute cycle is a good way to help students visualise what is happening inside the computer. For example, divide students into groups of four. Student A is responsible for generating the 'input' (i.e. a message or instruction for a task to be done), student B 'fetches' the instruction from student A and hands it to student C whose job it is to 'decode' the information and instruct student D to carry it out. Student A represents 'input', Student B the 'fetch' part of the cycle, Student C the 'decode' and student D 'execute'. You can increase the complexity of this task by having more than one person responsible for executing different types of instructions. For example, one student for something written, another for something spoken, and a third for a physical action (each representing a different output device). It would then be the job of Student C to decide which of these gets the correct message. The same could be done for input.
- Ask students to list as many storage devices as they can think of and then next to them detail what they are typically used for, their capacity and characteristics.
- What are the similarities between the human brain and a computer processor in the way they carry out instructions/tasks?

Enrichment activities

- Give students access to a physical computing device such as a Raspberry Pi, Arduino Board, Galileo Board, etc. Students should investigate and identify the key components on the board and what each element does. Can they identify where the CPU is? How does the device deal with memory or communicate with the other hardware devices attached to it? Ask students to set up a simple circuit with a single input and output. For example, when a motion sensor is activated, it results in a LED lighting up.
- Investigate new and upcoming technologies related to computer processing and data storage. What are the latest developments in computer memory going to be over the next five years?
- Ask students to investigate: What is Moore's law? Can the law continue as predicted? What developments will need to happen in technology in order for memory capacity to continue to increase as predicted?
- Investigate the timeline of how computer systems have developed over the years from the world's first computers, to today's technology and looking forward to future developments. What are the key things that have changed and why?

Assessment ideas

- Divide the class into groups. Assign each group a different concept from the list below:
  - Secondary storage
  - Fetch-decode-execute cycle
  - RAM and ROM
  - Central Processing Unit and its components
  - Computer systems and embedded systems.
- Each group needs to carry out in depth research into their chosen area to create a short interactive presentation to deliver to the rest of the class. The group's presentation should interact with the class audience and they might choose to 'teach' their peers or set a quiz. They should also create a digital message to highlight their chosen concept. This may be an animation, a program or other such media file. Each group has the opportunity to present their work to the class, and each group should participate in a Q&A session.

© Cambridge University Press

Answers

Activity 11.1

(a) Embedded devices have been built for a specific and limited purpose. All the components of the system are on a single circuit board. The memory contains the program and the board is contained within a larger device.

(b) Washing machine, dishwasher, elevators, fridges, coffee maker, navigation systems, etc.

Activity 11.2

Students' own answers

Activity 11.3

(a) ROM: is programmed to perform a specific function when it is manufactured, the BIOS is stored in ROM and that controls what happens when the computer starts up.

(b) RAM: temporarily stores programs that are currently in use so that they can be retrieved by the CPU quickly.

(c) Two differences between RAM and ROM are that RAM is volatile and all data stored within it is lost when the computer is turned off; whereas data within ROM is permanent. The computer cannot write to a ROM chip whereas it can write to the RAM chip.

Activity 11.4

(a) ALU – arithmetic and logic unit

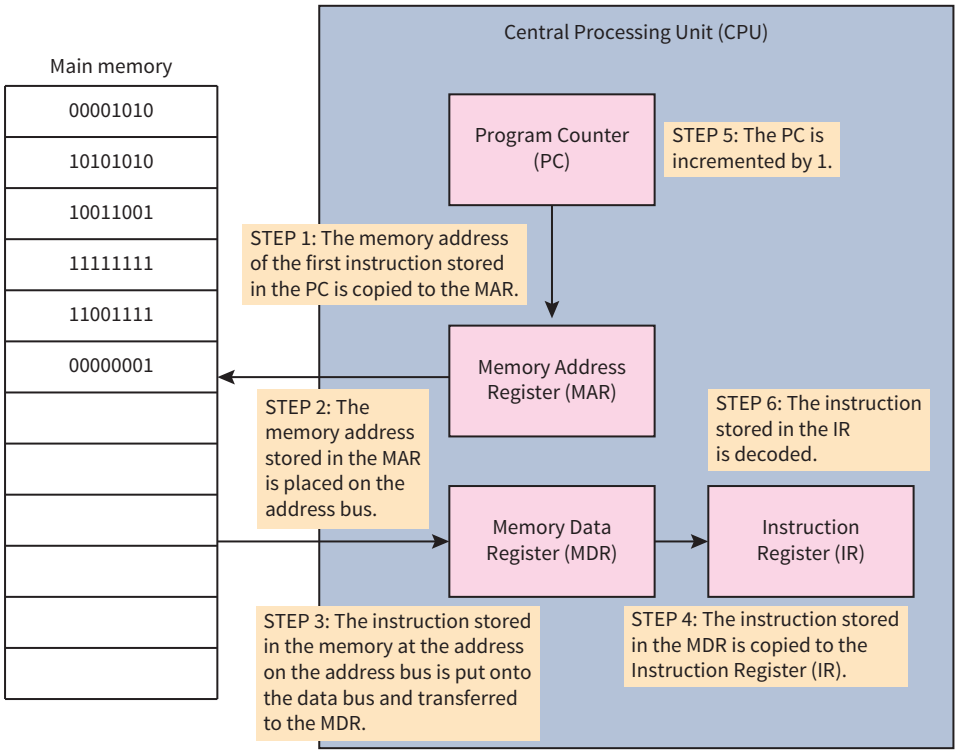
(b) Registers

(c) Control Unit

© Cambridge University Press

Activity 11.5

The diagram(s) should illustrate the events that take place during the fetch-decode-execute cycle and the role played by the components of the CPU. Some students might prefer to produce a list of steps rather than a set of diagrams (see below).



The fetch and decode part of the cycle

At the start of the fetch-decode-execute cycle the Program Counter (PC) holds the address in memory of the first instruction to be fetched from random access memory (RAM).

The address stored in the PC is copied into the Memory Address Register (MAR).

The address stored in the MAR is placed on to the address bus. The Control Unit (CU) issues a read signal and the instruction stored at that memory address is put onto the data bus.

The instruction on the data bus is loaded into the Memory Data Register (MDR), which acts as a temporary store (buffer) for anything that is copied from memory ready for the CPU to use.

The instruction in the MDR is copied to the Instruction Register (IR).

The PC is incremented by 1.

The CU decodes the instruction stored in the IR.

Execute

The CU carries out the instruction using the Arithmetic Logic Unit (ALU) for instructions involving arithmetic and logic operations.

Once the instruction has been executed, the cycle is repeated.

© Cambridge University Press

Activity 11.6

(a) overclocking

(b) caused due to increase in heat that can result in instability and damage to the processor. The fan is having to work harder to dissipate the heat.

Activity 11.7

(a) quad-core refers to four cores within the processor

(b) it is far quicker to retrieve memory from cache than RAM, so a cache means more data can be stored here and therefore be retrieved faster and thus improving performance.

Activity 11.8

(a) cloud storage so that people at home can see them.

(b) magnetic storage can be used as it is able to store vast amounts of data, with data being backed up continually throughout the day. Or constant backups can be made by syncing with a cloud server.

(c) USB solid state – as its portable and durable and easy to transfer between school and home.

(d) optical storage - for example, onto a DVD. Can be easily stored off-site.

© Cambridge University Press

Chapter 12: Computer systems: systems software

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain what is meant by systems software
- explain what is meant by an operating system
- describe the functions of the operating system
- explain what is meant by utility systems software
- list some examples of utility systems software and their functions.

What your students need to know

Students should:

- have knowledge of computer hardware.

Vocabulary

- Operating system
- Memory management
- Process management
- Peripheral management
- File management
- User interface
- Graphical user interface
- Utility systems software
- Full backup
- Incremental backup

Common misconceptions and other issues

Students sometimes do not appreciate that Windows is an operating system and its main function is to manage the operation of the computer and how it communicates with hardware. This is probably because windows usually comes 'bundled' with applications, such as a calculator, and games, such as solitaire.

Skills and coding

Coding skills:

- Use of arrays
- Use of a programming language to create and test a program to simulate disk defragmentation.

© Cambridge University Press

Skills and coding for non-specialist teachers

The final challenge requires the students to create and code an algorithm to simulate defragmentation.

The solution, in Python, is given below.

Python	Explanation
disk = [['C', '1'], ['', ''], ['A', '2'], ['B', '3'], ['C', '4'], ['C', '2'], ['', ''], ['A', '3'], ['', ''], ['B', '1'], ['', ''], ['B', '2'], ['C', '3'], ['C', '5'], ['A', '1'], ['', '']]	A two-dimensional array (list) containing the sectors as shown in the figure for 'Your final challenge' in the Student Book.
letters = ['A', 'B', 'C']	An array (list) to hold the three file names.
change = 0	A variable to hold the sector that has to be swapped.
for letter in range (0, len(letters)):	The loop will go through the letters A, B and then C.
number = 1	The variable 'number' is set to 1. The first search item will therefore be A1.
swapped = 1	The variable 'swapped' signals if a swap has been made.
while swapped == 1:	The 'while' loop will run while a swap has occurred – i.e. while swap is equal to 1.
swapped = 0	The 'swap' variable is now set to 0. It will be changed back to 1 if a swap occurs.
for index in range(0, len(disk)):	The 'disk' array is now searched.
if disk[index][0] == letters[letter] and disk[index][1] == str(number):	If the array contains an entry with the first data item equal to the letter and the second equal to 'number' then it is swapped with the location at index 'choice'. In the first loop, A1 will be swapped with the items at index 0.
temp = disk[index]	This code carries out the swap.
disk[index] = disk[change]	
disk[change] = temp	
number = number + 1	'number' is now incremented, for example, from 1 to 2. Therefore in the second turn of the loop, A2 will be searched for.
change = change + 1	'change' is incremented so that the swap will occur with the next index of the array.
swapped = 1	If a swap has occurred, the variable 'swap' is changed to 1 so that the 'while' loop will turn again. If 'swap' remains at 0, then the while loop will not turn and the next letter will be searched for.
print(disk)	The array is now printed with the items in their sorted positions.

© Cambridge University Press

Prompting questions

- What is software?
- Windows 95, Linux, iOS, Android, Windows 10 are all examples of software.
  - What do they have in common?
  - What type of software are they?
  - Can you think of any others that fit into this category?
- Why are backups important?
- Do you use any systems utilities on your home computers?
  - Which ones do you use most?
  - When do you use them?
  - Why do you use them?

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- Ask students to list the different types of software that they can think of. Encourage them to try and think of a diverse range of applications. Then ask students to move into pairs or groups of three and pool their lists together. Through discussion, can they group the software into different categories? What would those categories be and why? Why are particular grouping methods chosen by the students? This activity is best carried out as a starter before students have learnt about different software application categories. Follow the activity with a discussion amongst the class. Did more than one group devise similar categories?
- Describe the factors that an IT systems manager will need to consider in order to select the most appropriate type of backup system for their company.
- In pairs, discuss: what is the role of a 'driver' and how does it work? Why is it important to have the correct drivers installed on your machine? What might happen if the correct drivers are not installed?

Enrichment activities

- Investigate the 'onion diagram' for operating systems. This diagram shows the relationship between the computer hardware, the different aspects of the operating system, software applications and the user. Students should research the variations on this diagram and the relationship between the components described and then create their own improved onion diagram to illustrate this relationship.
- Investigate the terms multi-user and multi-tasking. What do these terms mean in relation to operating systems? Can students find examples of computers systems that are:
  - multi-user only
  - multi-tasking only
  - multi-user **and** multi-tasking
  - neither multi-user **nor** multi-tasking?
- Investigate different types of user interfaces. Find an example of each, and for each example state the software which uses that interface and discuss why this is/isn't appropriate for that application.

© Cambridge University Press

Assessment ideas

- Use the animation tools in a software package such as PowerPoint to create a simple animation to illustrate the concept of:
  - why a fragmented hard drive can slow a computer's performance
  - the role of the BIOS
  - the role of drivers
  - the importance of backups.

Students choose or are assigned one or more of the above.

© Cambridge University Press

Chapter 13: Networks

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- explain what is meant by a computer network and list the different types of networks
- describe the differences between client-server and peer-to-peer networks
- explain the functions of the hardware needed to connect computers
- explain how computers communicate using cable and microwave
- describe network topologies
- explain how users connect to and use the internet
- explain how data is transmitted across networks
- explain the use of protocols
- explain how virtual networks can be set up.

What your students need to know

- This chapter does not involve any coding and no specific skills or knowledge are required.

Vocabulary

- Network
- Personal area network (PAN)
- Local area network (LAN)
- Wide area network (WAN)
- Peer-to-peer network
- Client-server network
- Network interface card (NIC)
- Media Access Control (MAC)
- Hub
- Wireless access point
- Switch
- Router
- Ethernet
- Protocol
- Microwave
- Bandwidth
- Wi-Fi
- Frequency
- Channel
- Topology
- Star topology
- Mesh topology
- Node
- Internet
- World Wide Web (WWW)
- IP address
- Domain name

© Cambridge University Press

- Packets
- Packet switching
- Transmission Control Protocol / Internet protocol (TCP/IP)
- Networking layers
- Virtual network
- Virtual private network (VPN)
- Cloud storage

Common misconceptions and other issues

Hubs and switches: hubs relay messages received from each computer to all of the others on a single network, whereas switches inspect the messages and relay them only to the intended recipients.

They can do this as they build tables recording the MAC addresses of each computer.

Wireless access points are similar to hubs in that they do not relay messages to specific computers.

Routers connect different networks and, like switches, they can direct messages as they inspect each message.

Ethernet and Wi-Fi are both suites (or families) of protocols for communication within a network. Wi-Fi can be thought of as the wireless equivalent of Ethernet.

Wi-Fi is only one standard for wireless communication. Others include Bluetooth, 3G and 4G.

Students are often confused over the difference between the internet and the World Wide Web.

It should be stressed that the internet is a huge wide area network that allows communication between computers. The WWW is one of the services that run on the internet and others include email, file transfer, instant messaging and chat rooms.

The WWW is a system of interconnected documents formatted in HTML.

Skills and coding

- Coding skills:
  - No specific coding skills are required.

Skills and coding for non-specialist teachers

The chapter does not require any skills or coding.

Prompting questions

- What is a network?
- Why is a network useful?
- When have you used a network?
- Who has a network at home? Why did you choose to set up a network? What benefits have you received?
- What is the difference between a Personal Area Network and a Local Area Network
- What is the difference between a MAC address and an IP address?
- In a school network, which device would be preferable, a Hub or a Switch? Why?
- How is information sent across the internet?
- What is the relationship between cloud computing and networks?
- How does cloud computing work?

© Cambridge University Press

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

Draw a quick diagram that illustrates what a network is. This is a quick activity and students should not need more than a minute or so to do a rough sketch. After they have their drawing, ask them to compare results. Are there any similarities or differences? Ask students to justify or explain why their diagram looks the way it does. This will help clarify any prior understanding or misconceptions about networking that students may have if used as a starter activity.

Find and describe an example of:

- Personal Area Network
- Local Area Network
- Wide Area Network.

Consider peer-to-peer and client-server networks. What structure do you think dominates the internet and why?

Networks Unplugged is a series of activities that teaches the concepts of networks without using computers. It includes activities asking students to role play packet switching and a library mystery hunt amongst others. Resources can be downloaded from: **www.cambridge.org/links/kotd4015**

Investigate common protocols and what they do.

Enrichment activities

- Using animation software (or the animation tools in a presentation software package), create an animation that demonstrates how a network is structured and in particular about how data is sent across a network.
- Ask students to write an essay on the following scenario: 'Imagine waking up tomorrow and you find that networks no longer exist. While computers exist, each machine is a standalone machine and incapable of directly communicating with another device. Networks do not exist in any form. Describe your day, how different would your life be?'
- Investigate the network in your school. What type of network is it? What topology does it use? Can you identify which hardware components are used in the computer suite?
- Taking the students on a 'school trip' to the server room to meet the IT manager works well with this topic. Invite the IT manager to explain the structure of the network, how it is set up and their daily responsibilities. Students can prepare questions beforehand to ask on the day.

Assessment ideas

- Ask students to draw and annotate a diagram of their home (or school) network. The diagram should be labelled to show the main hardware components used and should illustrate network structure. Each student should be given the opportunity to present their network diagram and explain it.

© Cambridge University Press

Answers

Activity 13.1

1 i) A PAN allows your personal devices to connect to each other. This might be your laptop and mobile phone or television. The connection is usually over only a few metres.

ii) A LAN allows computers across a site to communicate with each other.

iii) A WAN allows computers across a large geographical area to communicate with each other.

2 In peer-to-peer networks, no computers have superiority over each other, whereas in a client-server network, one computer (server) provides resources to the client computers where users work.

Activity 13.2

Each computer will require a network adapter so that the computer can connect to the network and send and receive data.

Routers allow different networks to connect together. Routers can have both cable and Wi-Fi connections so will allow the business to have users connected via either medium and will allow the user on this network to connect to the internet.

Activity 13.3

Benefits and drawbacks using cable:

- Cables allow for higher bandwidth, so are great for media streaming
- Setup is much more difficult; hiding cabling so it doesn't become a safety hazard will need planning
- Better security, you need to be physically plugged in
- Not as portable since you need to be directly plugged in, for example you can't work from your garden unless you have a cable.

Benefits and drawbacks using Wi-Fi:

- Usually lower bandwidth than cable
- Easy and cheap to set up, you just need WAPS
- Signal can be affected by interference and distance so you might lose your Wi-Fi signal or data transfer slows
- Very portable, you can work from anywhere.

Activity 13.4

Benefits of using a star topology:

- If one computer fails, the other devices on the network will be unaffected and will carry on working.
- Adding or removing devices is easy and can be done without affecting the entire network.
- Data packets can be directed to the intended node directly without having to pass along the complete network. Consequently, there is less network traffic and fewer collisions.

© Cambridge University Press

Chapter 14: System security

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- describe the different strategies that criminals use to attack computer networks
- explain how people are the greatest security risks to networks
- describe the threats posed to networks
- explain how these threats can be identified, prevented and combatted
- explain the role of network policies.

What your students need to know

Students should:

- be able to use a programming language to create a menu-based information system about the security risks faced by computer users and how they can be avoided.

Vocabulary

- Social engineering
- Blagging
- Phishing
- Shouldering
- Pharming
- Malware
- Virus
- Worm
- Trojan
- Spyware
- Adware
- Brute force attack
- Denial of service
- Data interception
- SQL injection
- Zero-day attack
- Input sanitisation
- Encryption
- Firewall
- Backup policy
- Penetration testing

Common misconceptions and other issues

The material in this chapter is very straightforward and should not lead to any misconceptions.

© Cambridge University Press

Skills and coding

- Coding skills:
  - The use of a programming language to create a menu-driven information system.

Skills and coding for non-specialist teachers

Final challenge

This will provide the students with an opportunity to create a structured program using functions called from within the main program. This is an opportunity to apply some of the skills mentioned in earlier chapters.

Prompting questions

- Describe a recent computer crime that you might have heard of in the news or amongst your social network.
  - Why was it a criminal act?
  - Could it have been prevented?
  - How?
- What is meant by the term 'network forensics'? What would this team of experts do? When would they be used?
- What is shouldering? Has anyone ever been a victim or perpetrator of this?

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- Ask students to research a recent news report describing a computer crime. They need to find out what happened and then in pairs/small groups discuss if it could have possibly been prevented. Can they discover which legal act is enforced here and why?
- Compare the school's acceptable use policy with one from another organisation (for example, this one from the University of Bath: **www.cambridge.org/links/kotd4016)**. Ask students to identify common features and differences.
- Think-pair-share. Think: of one thing that you have understood well during the lesson and one question that you would like answered. Pair: share your question and area of confidence with your partner, can you answer each other's questions? Share: your common areas of confidence and any unanswered questions. Can someone else in the class answer it?
- Evaluate the security of your own devices and home network. What strategies do you employ? Could you improve the security? *Should you* improve security?
- What possible threats might you face with your personal devices and home network? Think-pair-share your thoughts and in groups discuss ways that these can be prevented.

Enrichment activities

- Research a criminal attack on a network. Prepare a two-minute presentation to deliver to the class describing the attack and important details surrounding it. What computer security features were breached?
- An excellent activity to explain public key encryption is available on CS Unplugged (**www.cambridge.org/links/kotd4018**). The activity is a good one to carry out with students, or the video listed below it also helps students to understand the nature of the process.

© Cambridge University Press

- Create a digital message that describes a computer crime/misuse and various ways to prevent these from happening.
- Write a program that will test the strength of a user's password upon entry. The program should output to the user whether their password is 'weak', 'medium strength' or 'strong'. It should also give appropriate suggestions about what can be done to improve the password strength.

Assessment ideas

- You have been given a training position with a network department of a new school that is just starting. Your task is to write a report outlining the range of risks that the school network will face and the security measures that should be put in place to prevent them.

Answers

Activity 14.1

(a) Phishing email

(b) Clue 1 – The email is not addressed to Catherine in person

Clue 2 – The writing style is careless, e.g. 'is terminated' rather than 'will be terminated'.

Clue 3 – Urgency, they want Catherine to respond within 24 hours or else.

Another clue is the inclusion of a link that Catherine is asked to click on to. This will almost certainly lead to a website controlled by the criminals.

Activity 14.2

A virus is a program that finds its way into a user's computer via another program or file. Once there it can attach itself to other programs and files. In contrast a user has to actively install a Trojan. They do this unwittingly either by opening an email attachment or by being fooled into thinking the software is legitimate. Both viruses and Trojans are harmful. They can corrupt data, delete files and – in the case of Trojans – enable criminals to access personal information including IDs and passwords.

Precautions users should take include:

- Installing firewalls to ensure software isn't downloaded without their knowledge.
- Keeping their computer's operating system up-to-date and installing the latest security patches.
- Installing anti-virus, adware removal and anti-spyware protection software.
- Avoiding opening emails and attachments from unknown sources.
- Only downloading programs from trusted websites and taking proper note of all security warnings, licence agreements and privacy statements

Activity 14.3

(a) A DOS attack is designed to grind a website or network to a halt by flooding it with useless communication. Criminals might use these attacks to extort money from the company to stop the attacks.

(b) Students' own answers.

(c) The foreign government might have wanted to prevent its citizens from accessing information it doesn't want them to see and to discourage the activists.

(d) Students' own answers.

© Cambridge University Press

Chapter 15: Ethical, legal, cultural and environmental concerns

LEARNING OUTCOMES

By the end of this chapter students should be able to:

- investigate and discuss the following issues in relation to the development and impact of computer science technologies:
    - environmental
    - ethical
    - legal
    - cultural
- discuss issues of data collection and privacy
- describe the legislation relevant to computer science.

What your students need to know

This chapter does not involve any coding and no specific skills or knowledge are required.

Vocabulary

- Environmental impact
- Data centre
- e-waste
- Ethical
- Lawful
- Digital divide
- Legislation
- Data protection act
- Computer misuse act
- Freedom of information act
- Copyright Designs and Patents Act
- Copyright
- Patent
- Creative commons licensing
- Proprietary software
- Open source software

Common misconceptions and other issues

Students often find difficulty in understanding the difference between ‘ethical’ and ‘legal’.

The students should be encouraged to discuss examples of situations which are legal but might not be considered ethical, for example: capital punishment, refusing to help an injured person.

Skills and coding

Coding skills:

- No specific coding skills are required.

© Cambridge University Press

Skills and coding for non-specialist teachers

No specific skills or coding are required for this chapter.

Prompting questions

- Would you prefer to take your exam on the computer or using paper? Why?
- What benefits do you think computing has made to your life?
- Computer legislation is often difficult to enforce. Why do you think this is the case?
- 'Technology has made the world a smaller place.' What do you think this statement means?
- Why do you think people prefer to use pirated copies of entertainment media rather than purchase the actual product?
- Is current computer legislation successfully tackling computer crime in the UK? If not, what do you think needs to change?
- What impact do you think wearable technology is having or will have on our lives?

Starters, plenaries, enrichment and assessment ideas

Starters and plenaries

- In groups, ask students to discuss how their lives today are different to the lives and childhood of their parents.
- Give students a short paper based quiz, then ask them to complete an online version (one can be chosen, or specifically designed using the many tools available). In groups, students should compare their experiences and explore the benefits and drawbacks of both approaches. This task should end with a class discussion summarising the differences.
- What are the 10 commandments of computer ethics? Can they be legally enforced? The website **www.cambridge.org/links/kotd4019** can be given to students if needed.
- What is Digital Rights Management?
- Write three key points about each legal act dealing with computer misuse.
- Write a poem / song / short story that discusses one of the computer laws or an ethical issue discussed in the chapter.
- Find examples of wearable technology. Choose two from your list and consider the potential benefits and drawbacks of using such technology.
- If you could write a quote about computers and the ethical, legal, cultural or environmental concerns surrounding it; then what would you say? Write your own quote. **Note to teacher: these student quotes, printed, would make a good display feature for the classroom if presented in the right way.**

Enrichment activities

- Speaking to older members of friends and family such as aunts, uncles, parents, grandparents, etc., find out what childhood was like for them.
  - What did lessons look like in school?
  - What did they do for fun?
  - How did they keep in touch with their friends?
- Compare these findings with your own lifestyle and answer the questions for yourself. This task can be extended by asking students to create a digital message highlighting the similarities and differences between the two age groups.

© Cambridge University Press

- Students should select and investigate one of the topics below. Students should consider both sides of the argument and end with their own opinion and justification of it. Their research findings should be summarised to be delivered as an interactive presentation to the class.
  - e-waste
  - Climate change
  - Energy production
  - Social communication
  - Drones
  - Security and surveillance.
- Computer piracy is a growing issue. A news story (**www.cambridge.org/links/kotd4020**) has highlighted the success of Netflix in Brazil, a country known for its widespread piracy issue. Read the news story. Why has Netflix been so successful? In groups, discuss what measures you think the entertainment industry needs to take to tackle and reduce computer piracy in the UK.
- The digital divide still exists in the UK. Investigate what the government intends to do to tackle it.
- Artificial intelligence has increased and improved significantly over recent years. Investigate examples of current technology that uses artificial intelligence. How many examples can you find? What impact do you think they have had on our lives?
- Wearable technologies are increasing and perhaps the most famous examples of wearable technologies are Google Glass and the Apple Watch. Yet the Apple Watch has been more socially acceptable than Google Glass. There was lots of excitement around the project before release, but this quickly dwindled when the product came to market. Google even released guidance for its users. Investigate the reviews and news reports around Google Glass. What were the issues and debates surrounding the technology? Why did Google have to undertake a review of the entire project?

Assessment ideas

- Write an essay on: 'Everyone needs to learn to code.' Discuss.
- Choose one of the quotes from the list below. Ask students to investigate the quote. What did they think the speaker was referring to? Do they agree with what was said? Using the quote as a guiding point, ask the students to write an essay to discuss the issues highlighted within the quote.
  - *Computers themselves, and software yet to be developed, will revolutionize the way we learn*. **Steve Jobs**
  - *Personally, I rather look forward to a computer program winning the world chess championship. Humanity needs a lesson in humility*. **Richard Dawkins**
  - *Home computers are being called upon to perform many new functions, including the consumption of homework formerly eaten by the dog*. **Doug Larson**
  - *I do not fear computers. I fear the lack of them*. **Isaac Asimov**
  - *Computer science is no more about computers than astronomy is about telescopes*. **Edsger Dijkstra**
  - *Security is, I would say, our top priority because for all the exciting things you will be able to do with computers - organizing your lives, staying in touch with people, being creative - if we don't solve these security problems, then people will hold back*. **Bill Gates**
  - *Every piece of software written today is likely going to infringe on someone else's patent*. **Miguel de Icaza**
  - *The Internet is not just one thing, it's a collection of things - of numerous communications networks that all speak the same digital language*. **Jim Clark**
  - *Supercomputers will achieve one human brain capacity by 2010, and personal computers will do so by about 2020*. **Ray Kurzweil**

© Cambridge University Press

Answers

Activity 15.1

This activity focuses on the role of computer scientists in combating global climate change. Students should avoid broadening out the discussion to include other ways in which computer science can help protect the environment, such as animal conservation or natural resource management.

They should begin by defining the term 'global climate change', that is, a long-term shift in weather conditions including temperature, rainfall and winds, and identifying factors that contribute to climate change – in particular the build-up of greenhouse gases caused by human activity – notably the burning of fossil fuels.

They should then describe a number of different ways in which computer scientists can play a role in combating climate change, such as:

- developing computer systems that combine satellite observations, ground-based data and forecast models to monitor and forecast changes in the weather and climate
- producing computer models to explore the impact of climate change on water resources, crop yields, fish stocks etc. and aid planning
- contributing to the development of renewable energy sources to reduce our dependence on fossil fuels
- developing smart systems for homes, offices, cars etc., that use energy more efficiently
- developing battery technology so that energy generated (when the sun is shining or on a windy day) can be stored and used when required.

They should find appropriate examples to support their discussion.

This activity could provide useful exam practice, demonstrating how to tackle a question that requires an extended, essay-style answer.

Activity 15.2

Students' own answers

Activity 15.3

This activity is designed to generate some lively classroom debate. There is no need for students to delve too deeply into the Luddite movement. Instead, the discussion should focus on the impact of new technology. For example, students could review the effects of automation on labour-intensive repetitive jobs in sectors such as car manufacturing, food processing and retail sales. They might also want to consider how advances in artificial intelligence and robotics could impact on 'professional' jobs in fields such as Medicine, Transport and Finance. This theme is picked up again in **Activity 15.8** which focuses on digital inclusion.

Activity 15.4

There are a whole host of ethical dilemmas associated with the growing use of robots that students might want to consider. These include issues associated with privacy, security and human dignity as well as the moral obligations of society towards its robots. A starting point for this discussion could be to investigate Isaac Asimov's 'three laws of robotics', drawn up in the 1940s.

Students might find it helpful to focus on a few specific examples, such as driverless cars, home robots, combat robots and robotic surgery.

Hopefully students will conclude that robotics is a good thing, with the potential to transform the way we live and work, but that it is important to put in place appropriate limitations and controls on their use.

© Cambridge University Press

Activity 15.5

This follows on from **Activity 15.4** and is designed to provoke further lively classroom debate. Students could investigate the codes of conduct issued by the three main professional bodies BCS, ACM and IEEE to see if they shed any light on this ethical dilemma. All three require their members to treat people fairly, honour contract agreements and adhere to company policy, which provides something of a steer in this case.

Students could try to find examples of serious software bugs and then put them in order of severity. This should provoke some interesting discussion. What algorithm did they apply to decide on the order? Should bugs that threaten human life always be dealt with first? What happens if there's a very slim chance of this happening?

Activity 15.6

This is another interesting topic for a class discussion. Students should begin by summarising what the Regulation of Investigatory Power Act (RIPA) was designed to do , that is, to govern the interception and use of electronic communications in order to ensure that the way investigatory powers are used by organisations such as councils and government departments complies with human rights law, in particular, the European Convention on Human Rights.

In response to the increased threat from terrorist organisations, the police and security services are calling for even greater powers to monitor internet traffic, mobile devices and other forms of electronic communication, as well as to extend the use of CCTV surveillance in public places. Students might want to consider whether this increased intrusion on privacy is justified. Do law-abiding citizens have anything to worry about? Should we be concerned that surveillance could be used for reasons other than those stated?

The RIPA was passed back in 2000. Since then, surveillance technology has moved on apace. Students might also want to investigate the implications of the draft Investigatory Powers Bill currently going through parliament – regarded by some as a snoopers' charter.

Activity 15.7

(a) Examples of how new technology impacts on the way in which people interact with one another could include:

- Individuals are more likely to use social networking sites, such as Facebook and Twitter, to socialise and keep in touch with friends and family members, than to interact with them face-to-face.
- Social networking sites and other online forums enable individuals to communicate with more people across greater distances faster than ever before. However, information, which might once have been shared privately, is now often published online for anyone to read and see.
- Social interaction within the family can suffer when individual family members are preoccupied with their smartphones, tablets, games consoles etc. Instead of spending time interacting with each other, they are engrossed in their own digital world.

(b) Examples of how new technology impacts on work and employment could include:

- Whilst demand for highly skilled workers is increasing, some jobs are disappearing as automated systems eliminate the need for human workers.
- Technology supports more flexible working practices, enabling people to work anywhere anytime, but potentially making it more difficult for them to maintain a healthy work-life balance.
- Technology makes it possible to outsource some jobs away from areas where wages are high and workers have well-established employment rights to developing parts of the world where employment costs and overheads are lower.

© Cambridge University Press

(c) Examples of how new technology impacts on education could include:

Textbooks – like this one – are no longer limited to merely text and pictures. They have links to additional web-based materials, including animations, videos, practice assessments etc. to support the learning of new content and make learning more interactive.

Online education and training is available to anyone anywhere, providing they have access to the internet.

It's much quicker and easier to find information online than it is to visit a library. But information overload can be a problem. Good information handling skills are required to sift through the huge amounts of information available online.

(d) Examples of how new technology impacts on leisure could include:

- Digital entertainment, including music, films and books is available on demand 24/7 via the internet.
- Multi-user online games enable users to play with other people from all over the world.
- Fitness gadgets such as the Apple Watch enable users to monitor their activities and motivate them to do more.

Activity 15.8

Chapter 15 stresses the benefits of new technology. The premise of this activity is that these benefits should be available to all, irrespective of wealth, gender, ethnicity etc. and that society should attempt to ensure that this is the case.

Students might want to summarise factors contributing to the digital divide and evaluate efforts being made by governments to promote digital inclusion.

Activity 15.9

Five reasons why data stored online is less secure than paper-based storage are listed below.

- It could be hacked.
- It could be maliciously damaged or destroyed by malware.
- It could be stored on servers belonging to a third party over whom the data owner has no control/ jurisdiction.
- It could be damaged or destroyed as a result of hardware or power failure or human error (though a sensible back-up and recovery regime should prevent this).
- Large-scale theft of digital records is much easier than stealing large quantities of paper records – thieves would need an articulated lorry to steal 10 000 paper records.

Five advantages of online storage over paper-based storage are listed below.

- Data stored online can be accessed from anywhere, providing there is an internet connection.
- Multiple people can have access to the same files simultaneously.
- In the event of a fire, flood or other disaster that affects your place of work, data stored online will remain unaffected.
- There's no need to set aside physical space to store paper files or to predict how much storage space is likely to be required in the future. Cloud storage is much more flexible, you only use and pay for the amount you need and can have more or less as and when required.
- Encryption can be used to add an additional layer of security to digital data stored online.

© Cambridge University Press

Activity 15.10

(a) intentional and unauthorised destruction of software or data

(b) unauthorised access with intent to commit further offences

(c) unauthorised access to computer material.

Activity 15.11

Students' own answers

© Cambridge University Press

Non-exam assessment (NEA)

The non-examined assessment component targets primarily Assessment Objective **AO3** but also **AO1** and **AO2**. It is **marked out of 40** and has a **weighting of 20%.**

OCR will issue three assessment tasks at the start of the terminal academic year of assessment. Only tasks designated for that examination series can be submitted unless carrying forward marks from a previous year.

Students must complete **one** of the tasks chosen by the school or themselves.

It requires them to analyse a problem and then design, implement, test, refine and evaluate a computer program to solve it and is assessed in the following areas: **Programming techniques (12 marks), Analysis (6 marks), Design (8 marks), Development (8 marks), Testing and evaluation and conclusions (6 marks).**

Students may complete the assignment over multiple sessions, up to a **combined duration of 20 hours.**

The program must be written in a suitable high level language. The specification contains a list of languages that **could** be used but **does not stipulate** that only one of those must be used.

All work submitted by a student must have been done under observation by their teacher and the final report must be only their own work. External sources can be used but must be referenced and no marks can be awarded for materials submitted which are not the student's own.

The students are not allowed to take the NEA tasks nor any work associated with them home and are not allowed to access their projects online form home.

The students must not be allowed to access email or storage locations to prevent them using work done outside of the supervised sessions.

Teachers cannot give detailed advice and suggestions as to how the work may be improved in order to meet the assessment criteria. This includes indicating errors or omissions and personally intervening to improve the presentation or content of the work.

The report should be submitted as a single document, preferably in .pdf format, and all the code must be visible in the report and be fully annotated.

The project should be marked according to the marking criteria using a 'best fit' approach. For each of the marking criteria sections, teachers must select one of the three band descriptors provided in the marking grid that most closely describes the quality of the work being marked.

Further details can be found in the NEA section of the Student Book.

© Cambridge University Press

Worksheets: Answers

ANSWERS

Worksheet 1.1

1 (a) An algorithm is a step-by-step procedure for solving problems. It is a set of instructions that can be followed by humans and computers.

(b) There will be many possible solutions, but the sequence should be correct.
This is an example answer.

Leave the house
Walk to the bus stop.
Wait for a bus to arrive.
If it is the correct bus, then get on.
Sit on the bus until it reaches the school.
Get off the bus.
Walk into school.

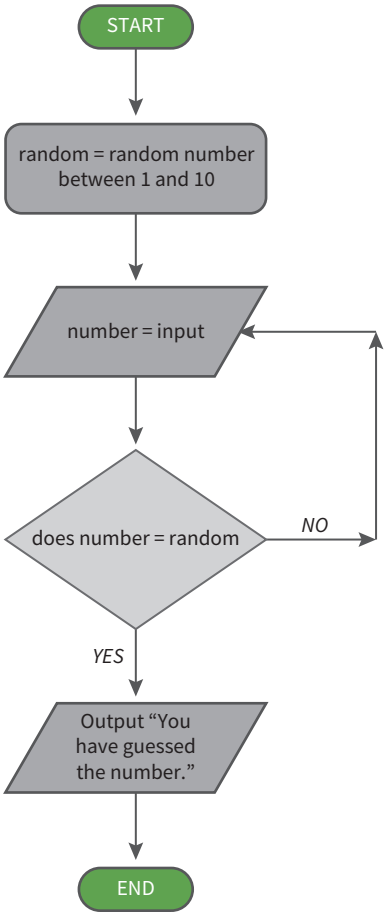
(c) Sequence

Worksheet 1.2

1 (a) A **selection** in an algorithm is where a question is asked, and depending on the answer, the program takes one of two courses of action.

(b) An **iteration** in an algorithm is where a task is repeated for a set number of times or until there is a required outcome.

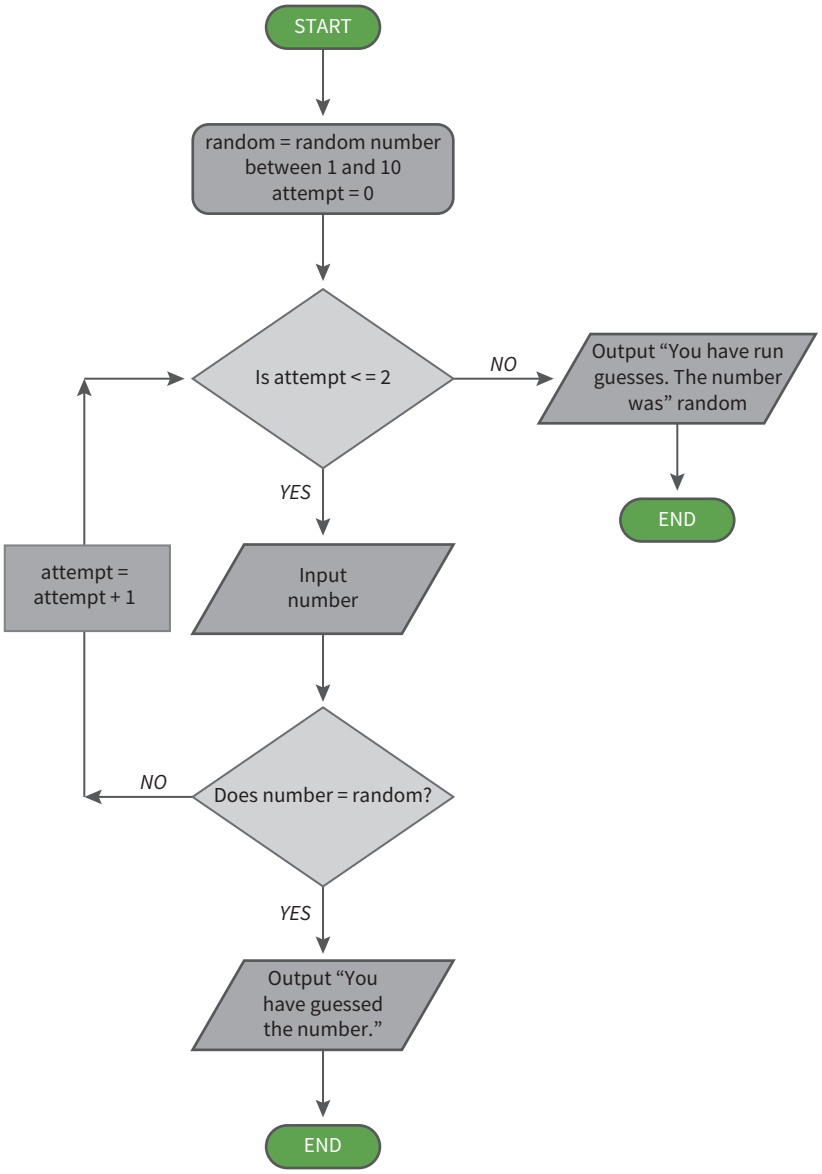
2



© Cambridge University Press

Worksheet 1.3

1



Worksheet 1.4

1 (a) Result = 7

(b) Result = 1

(c) Result = 27

2 (a) (i) True

(ii) False

(iii) True

(b) (i) false

(ii) true

(iii) false

© Cambridge University Press

Worksheet 1.5

1

```
number1 = input("Please enter the first number.")
number2 = input("Please enter the second number.")
number3 = input("Please enter the third number.")
if number1 > number2 then                    // number1 is compared with number2
       if number1 > number3 then             // if it is greater, then it is
                                              compared with number3
              highMark = number1             // if it is greater, then it must be
                                             the highest
       else
              highMark = number3             // if not then number3 must be
                                             the highest
       endif
elseif number2 > number3 then                // if number2 is greater than
                                             number1 it is compared with
                                             number3
       highMark = number2                    // if it is greater then it must
                                             be the largest
else
       highMark = number3                    // if not then number3 must
                                             be the largest
endif
print("The largest number is " + highMark)
```

Worksheet 2.1

1 (a) An **iteration** in an algorithm where a task is repeated for a set number of times or until there is a required outcome.

(b) A **definite iteration** is when the number of iterations is known before the execution of the loop is started, for example, it may be set to three or five times and it will execute that number of times, whatever the conditions, unless there is a command to break out of the loop.

2

```
       number1 = input("Please enter a number.")
       number2 = input input("Please enter a higher number.")
       for index = number1 to number2
              print (index)
       next index
```

© Cambridge University Press

Worksheet 2.2

1 (a) In **indefinite iteration** (also known as conditional iteration), the number of iterations is not known before the loop is started. The iterations stop when a certain condition becomes true or false.

(b)

```
numberEntered = 0
total = 0
reply = "y"
while reply == "y"
        number = input("Please enter a number.")
        numberEntered = numberEntered + 1
        total = total + number
        reply = input("Enter 'y' to enter another number or 'n' to stop.")
endwhile
print("You entered " + numberEntered + " numbers and the total is " + total)
```

Worksheet 2.3

1 (a)

```
// Nested loops are used to check all of the possible combinations
for numberCoffees = 1 to 100
        for numberTeas = 1 to 100
                total = (numberCoffees*1.9) + (numberTeas*1.2)
                if total == 285 then // if the total equals 285 the
                                     number of teas, coffees and total
                                     are stored
                        teas = numberTeas
                        coffees = numberCoffees
                        totalTaken = total
                endif
        next numberTeas
next numberCoffees

print(totalTaken)
print(teas)
print(coffees)
```


© Cambridge University Press

Worksheet 2.4

1

Array				Index	Total	Output
3	9	6	13	0	3	
3	9	6	13	1	12	
3	9	6	13	2	18	
3	9	6	13	3	31	31

2

a	B	c	Output
10	0	0	
9	1	8	
8	2	14	
7	3	18	
6	4	20	
5	5	20	
4	6	18	
3	7	14	
2	8	8	
1	9	0	1, 9, 0

Worksheet 3.1

1 (a) AND gate

(b) NOT gate

(c) OR gate

2

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Worksheet 3.2

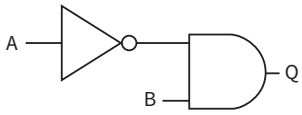
1

A	B	C	D	E	Q
0	0	0	1	0	1
0	0	0	1	1	1
0	1	0	1	0	1
0	1	0	1	1	1
1	0	0	1	0	1
1	0	0	1	1	1
1	1	1	0	0	0
1	1	1	0	1	1

© Cambridge University Press

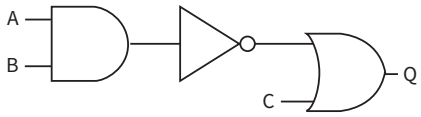
Worksheet 3.3

1



A	B	Q
0	0	0
0	1	1
1	0	0
1	1	0

2



A	B	C	Q
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Worksheet 4.1

1 (a) An **integer** is a whole number without decimals (can be positive or negative).

(b) A **real** number is any number that exists including their decimals and fractions.

(c) A **string** is a list of characters of any length. It can include alphanumeric data and symbols.

2 (a) Boolean

(b) String

(c) Integer

(d) Integer

Worksheet 4.2

1 In the algorithm, there are two logical errors. For each, state the line number and how the error could be corrected.

Line number: 1

Correction: length = string.length – 1

Line number: before line 3

Correction: found = false

while found == false AND index <= length

© Cambridge University Press

Worksheet 4.3

1 An **array** is a data structure that contains many items of data of the same type under the same variable name. The data is indexed so that a particular item of data can be easily found

2 (a) [1,1]

(b)

```
productName = input("Please enter the name of the product.")
for index = 0 to products.length -1
        if products[index, 0] == productName then
                productPrice = products[index, 1]
        endif
 next index
 print("The price of" + productName + "is" + productPrice)
```

Worksheet 5.1

1 (a)

Pass 1						
Jackson	Brown	Morrison	Andrews	Fleming	Smith	Wilkinson

Pass 2						
Brown	Jackson	Andrews	Fleming	Morrison	Smith	Wilkinson

Pass 3						
Brown	Andrews	Fleming	Jackson	Morrison	Smith	Wilkinson

Pass 4						
Andrews	Brown	Fleming	Jackson	Morrison	Smith	Wilkinson

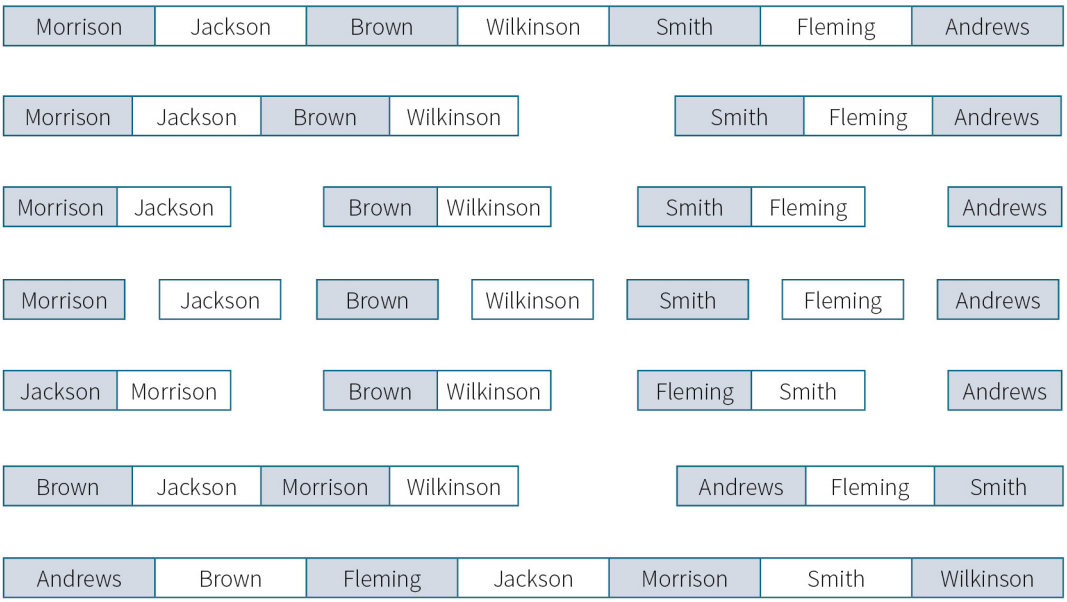
(b) Another pass is needed as the algorithm must continue until there are no swaps in a pass and in the next pass, there will be no swaps.

Worksheet 5.2

1 A **merge sort** algorithm is used to sort an unordered list by repeatedly (recursively) dividing a list into two smaller lists until the size of each list becomes one. The individual lists are then merged with the items in the correct order.

© Cambridge University Press

2



Worksheet 5.3

1 A **linear** search algorithm is a simple, sequential search. It starts at the beginning of the list and moves through the items, one-by-one, until it finds a matching value or reaches the end without finding one.

A **binary** search algorithm can only search an ordered list to find an item by looking at the middle (median) item and comparing it with the search value. If the search item is smaller than the median then the median of the sublist to the left is searched and vice versa, if it is larger. This continues until the search item is found or there are no items left to search.

2 (a) For a **linear** search, the best case would be 1 selection if the search item was at the start of the list and 1000 if it was the last item.

(b) For a **binary** search, the best case would be 1 if the search item was the median and 10 for the worst case as the following medians could be chosen – 500, 250, 125, 63, 32, 16, 8, 4, 2, 1.

Worksheet 5.4

1

```
The item to be searched for is stored in the variable 'item'.
      low = 0                                  // 'Low'  is the first item in
                                               the array named 'list'.
      high = list.length - 1                   // 'High'  is the last item in
                                               the list.
      found = False
      while low <= high AND found == False     // The search will continue while
                                               there are items in the list and the
                                               search item has not been found.
            midpoint = (low + high)/2          // The median is found.
            if list[midpoint] == item then
                  found = True
            elseif item < list[midpoint] then
```


© Cambridge University Press

```
                high = midpoint-1          // If the item is lower than
                                           the median, then the sublist
                                           to the left is searched.
        else
                low = midpoint+1           // If the item is higher than
                                           the median, then the sublist
                                           to the right is searched.
        endif
endwhile
if found == True then
        print("The item is in the list.")
else
        print("The item is not in the list.")
endif
```

Worksheet 6.1

1 **Validation** is the process through which the program checks that data is sensible, reasonable and appropriate to be processed by the program.

(a) A **range check** makes sure that the data is in a certain range or falls between two points (e.g. is less than 20 or is between A and F).

(b) A **length check** ensures that the data has a certain number of characters (e.g. that a password has a minimum of eight alphanumeric characters).

(c) A **presence check** is usually used when users have to enter data, ensuring that they have actually entered something and that the data is present.

Worksheet 6.2

1 A **modulus check digit**, like all check digits, is the help prevent transcription errors which occur when users enter data into a computer. The check digit ensures that the data entered could be valid but it does not ensure that it is the data that should have been entered.

2

3	9	6	3	5	6	9

Multiplied by weighting:

24	63	36	15	20	18	18

Total = 194

Divided by 11 = 17 Remainder 7

Subtract remainder from 11 = 4

Therefore modulus check digit = 4

Complete 8 digit number = 396 35694

© Cambridge University Press

Worksheet 6.3

1 The array is **friends.**

```
myFile = openWrite("Friends.txt")
    for index = 0 to friends.length - 1     // The loop could also be 0 to 6
                                            as there are seven items.
        myFile.writeLine(friends[index])
    endfor
myFile.close()
```

Worksheet 7.1

1 **Decomposition** means breaking a problem down into smaller, more manageable parts which are then easier to solve. It is an example of the 'divide and conquer' approach to problem solving.

2 Decomposition of the problem should identify some of the following sub-problems:

- Design of interface showing the 3×3 grid.
- How to keep track of which squares have been selected by 'X' and '0' and which are free.
- How the 'computer' will decide which square to select when it is its turn.
- How the 'computer' will decide when the game is over.
- How the 'computer' will decide who has won a game and keep count of the score.
- How the user can ask for another game or quit the program.

Worksheet 7.2

1 **Abstraction** is the process of removing unnecessary details so that only the main, important points remain. When creating an algorithm to model a real-life action or activity, abstraction identifies essential elements that must be included in the computer models and discards inessential ones.

2 (a)

```
function dice()                     // This is the function definition.
                                    No parameters are passed to it.
    dice1 = random(1, 6)            // This generates a random number
                                    between 1 and 6.
    dice2 = random(1, 6)
    totalDice = dice1 + dice2
    return totalDice
endfunction

score = dice()                      // This is the statement in the main
                                    program which calls the function.
                                    No arguments are passed to it.
```


© Cambridge University Press

(b) The important feature or property of a dice is that it generated a random number between 1 and 6. This is the only property that needs to be represented in the model. Other features such as colour or the material it is made from are unimportant. Abstraction has been used to identify the only relevant property of the dice, i.e. that it has generated a random number between 1 and 6.

Worksheet 7.3

1 A **subroutine** is a self-contained module of code that can be 'called' by the main program when it is needed.

2 **Functions** are called from an expression in the main program and return a value to it. Procedures are called from a statement in the main program but do not return any value to it. They just do something and then the main program continues.

3

```
procedure finalPrice(price)                    // This defines the function with
                                               the parameter 'price'.
        if price > 200 then
                payment = price - (price/100*10)
        elseif price >= 100 AND price <= 200 then
                payment = price - (price/100*5)
        else
                payment = price
        endif
                return payment          // Payment is passed back to the
                                        main program where it is stored in
                                        the variable 'customerPayment'.


endfunction


cost = total of all the goods bought by the customer.
customerPayment = finalPrice(cost) // 'Cost' is passed to the function
                                        as an argument. In the function, it is
                                        the parameter 'price'.
```

Worksheet 7.4

1 The answer should contain the following points:

- Subroutines are a natural way of implementing computational thinking because some of the tasks identified can be allocated to a subroutine. They therefore assist with decomposition and abstraction.
- Repeated sections of code need only be written once and called when necessary.
  - This shortens the development time of a program and means that the finished program will occupy less memory space when it is run.
- Subroutines improve the structure of the code, making it easier to read through and follow what is happening.
- It's easier to check the program because each subroutine can be coded and tested independently.

© Cambridge University Press

- The program is easier to debug as each subroutine can be inspected independently.
- If changes have to be made at a later date, it is easier to change a small module than having to work through the whole program.
- In large development teams, different members can be working independently on different subroutines.
- Standard libraries of subroutines can be built up and they can be reused in other programs.

Worksheet 8.1

1 The microprocessor contains the central processing unit, which carries out all of the program instructions by carrying out millions of calculations each second.

These calculations are performed by billions of transistors acting as switches. They are either 'on' or 'off'. They have only two states: they either transmit an electric current or they do not.

Any system involving two states is called a binary system.

As there are only two states ('off' or 'on'), they can be represented by the two digits of the binary system: 0 and 1.

All computer programs are lists of instructions switching transistors 'off' or 'on' and therefore they can be represented by the digits 0 and 1.

2

Decimal prefix is used.

Unit	Number of bits
Megabyte	8 000 000
Nibble	4
Byte	8
Terabyte	8 000 000 000 000
Kilobyte	8 000
Gigabyte	8 000 000 000

3 213 624 133 bits = 26 703 016.6 bytes

= 26 703 kilobytes

= 26.7 megabytes

Worksheet 8.2

1 (a)(i) 213

(ii) 110

(iii) 170

(b) (i) 11101001

(ii) 10011001

Worksheet 8.3

1 An **overflow error** occurs when a calculation produces a result that is greater than the computer can deal with or store.

© Cambridge University Press

2 (a)

1	0	0	1	1	1	1	0
0	1	0	1	0	0	1	0
1	**1**	**1**	**1**	**0**	**0**	**0**	**0**

(b)

	0	1	1	1	1	1	0	1
	1	1	0	0	1	1	1	1
1	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**

(c)

0	0	1	0	1	1	0	0
1	**0**	**1**	**1**	**0**	**0**	**0**	**0**

Worksheet 8.4

1 Computers do not understand or use hexadecimal; they only understand and use binary.

Hexadecimal is used by computer scientists because people get confused with large binary numbers, so we simplify binary numbers by representing them in hexadecimal notation, meaning that fewer numbers are used.

2 (a) DD

(b) 92

3 11001101

Worksheet 9.1

1 The character set is the complete list of binary codes that can be recognised by computers as being usable characters.

2 (a) C

(b) a

3

```
for index = 0 to myString.length - 1
      print(myString(index) + " " + ASC(myString(index)))
next index
```

Worksheet 9.2

1 (a) The **image size** is the number of pixels in the width and then the number of pixels in the height, for example 640 × 480 or 2048 × 1536.

(b) The **image bit depth** or colour depth is the number of bits used to encode the colour of each pixel in an image. The greater the number used, the greater is the image quality.

2 Image size = 4096 × 2480 × 24 = 243 793 920 bits = 30.4 megabytes.

3 The **sample rate** is the number of samples of the sound taken each second. The higher the sample rate, the more accurately the sound will be represented. A sample is a physical measurement of the sound recorded as a binary value.

4 File size = 180 × 2 × 44 100 × 16 = 254 016 000 bits = 31.7 megabytes

© Cambridge University Press

Worksheet 9.3

1 (a) During lossless compression, no data is lost and the file can be decompressed with all its information intact.

(b) During lossy compression, data is lost in the compression process and when the file is decompressed, it will not contain all the original information

2 A lossless compression algorithm would more successfully compress a black-and-white image file than a true colour one because there are long runs of only two colours, either black or white, but a colour photograph has short runs of many different colour variations (16 777 216).

3 wwwwwwbbbwwwwwwbbbbbbwwbbbwbbbwbbwbbb = 6w3b6w6b2w3b1w3b1w2b1w3b

Worksheet 10.1

1 The **instruction set of a computer** is the set of instructions for its particular processor that it can understand and is able to process.

2 **Machine code** or **machine language** presents the instructions in a form that the processor can execute; strings of 0s and 1s.

3 (a) **Assembly languages** are called **low level languages** because they are very similar to machine code in concepts and structure. There is a one-to-one relationship between the commands in assembly language and those in machine code. Assembly language is at a low level of abstraction from the machine language.

(b) The binary opcodes which are binary in machine code are represented as descriptive words called **mnemonics** in assembly language (e.g. SUB, MUL and DIV). These are easier for humans to remember and use.

4 An **assembler** translates the mnemonics of assembly language programs into machine language instructions for the microprocessor to execute.

Worksheet 10.2

1 (a) and (b) Answers could include:

- Faster program development: it is less time consuming to write and then test the program.
- It is not necessary to remember the registers of the CPU and mnemonic instructions.
- Portability of a program from one machine to other: each assembly language is specific to a particular type of CPU, but most high-level languages are generally portable across different CPUs.

2 (a) Programs written in high-level languages (source code) must be translated into machine code (object code) before the processor can execute them as it can execute instructions only if they are presented in machine code.

(b) (i) A **compiler** translates the source code into a standalone, machine code program (object code) which is output as a new file that can then be executed by the processor.

An **interpreter** translates the high-level code line-by-line into machine code each time it is run.

(ii) **Benefits of compiler:**

- The program that is run is already translated into machine code, so it can be executed more rapidly.
- It protects the software from competitors who would otherwise be able to see the source code.
- If it encounters any errors, it carries on.

© Cambridge University Press

Benefits of interpreter:

- When an error is found, the interpreter reports it, stops and pinpoints the error, so that the programmer knows where it has occurred.
- The code is not platform-specific and can be run on different operating systems and platforms if there is an interpreter.
- The program can be easily edited as it always exists as source code.

Worksheet 11.1

1 The **hardware** consists of the physical devices of a computer system, such as the keyboard, processor and storage devices.

The **software** consists of all the programs that the hardware uses to operate.

2 (a) An **embedded system** is a computer system built into another device in order to control it. The components are on a single printed circuit board.

(b) It monitors the water temperature so that it can turn the heating element on and off to maintain the correct temperature.

(c) Any of these three are suitable: television, microwave, cooker.

Worksheet 11.2

1 (a) The **control unit** co-ordinates the actions of the computer by sending out control signals to the other parts of the CPU, such as the ALU and registers, and to the other components of the computer system, such as the input and output devices.

(b) The **arithmetic and logic unit (ALU)** performs arithmetic and logical operations. It carries out activities such as, addition and subtraction, multiplication and division, logical tests using logic gates and comparisons such as whether one number is greater than another.

(c) **Registers** are storage locations within the CPU itself. They can be accessed even more quickly than the main memory. Some of these registers serve specific functions, but some of them are general purpose registers used for the quick storage of data items.

(d) **The clock** regulates the timing and speed of all computer functions. Within the clock is a quartz crystal which vibrates at a particular frequency when electricity is applied to it. Pulses are sent out to the other components to co-ordinate their activities and ensure instructions are carried out and completed. One instruction can be carried out with each pulse of the clock, and therefore the higher the clock rate, the faster the CPU will be able to carry out the program instructions.

Worksheet 11.3

1 (a) A **quad core processor** has four core processing units within the CPU.

The advantages of multiple core processors over single core processors are:

- the cores can work together on the same program; this is called parallel processing.
- the cores can work on different programs at the same time; this is called multitasking.

However, not all programs will run at four times the speed with a quad-core processor. The tasks required might not be able to be carried out in parallel. They might be sequential so that one task requires output from a previous task, so the second task cannot start until the first has finished.

(b) **L1, L2** and **L3** cache are memory modules consisting of fast dynamic RAM within or very close to the CPU. The fastest is the Level 1 cache and is smaller than the Level 2 and Level 3 caches.

The caches speed up the processing speed of the CPU by storing recently used data and data likely to be frequently used so that so that data does not have to be collected from the slower RAM when it is

© Cambridge University Press

needed. The L1 cache is checked first followed by the L2 and then L3 caches. In a multi-core processor, the cores have their own L1 and L2 caches.

Worksheet 11.4

1 Secondary storage devices are necessary because RAM is volatile. Data is stored on devices called secondary storage devices so that it is not lost when the computer is switched off. The secondary storage devices also allow data to be transferred between devices.

2

Use	Magnetic, optical or solid state	Reason why this is the most appropriate
Storing images in a digital camera	Solid state	It is light and as there are no moving parts; it is not damaged when devices are knocked or dropped.
Storage devices on a school's main fileserver	Magnetic	Store large amount of data and are relatively cheap Fast data access speed
Videos of a school production to be given to parents	Optical	Cheap and easy to transport data between locations
Handheld devices used by students for fieldwork	Solid state	It is light and as there are no moving parts; it is not damaged when devices are knocked or dropped.

Worksheet 12.1

1 (a) The **memory manager** is in charge of the RAM. Programs often need to use the RAM throughout their operation. Some programs will use the RAM extensively whereas other smaller programs will use it less frequently. The memory manager checks that all requests from programs for memory space are valid and allocates them accordingly.

(b) The **peripheral manager** controls all the computer input and output by managing requests from programs to use devices such as printers, speakers, keyboards and hard disk drives.

The peripheral manager communicates with the devices through software called 'drivers', which translate the instructions sent by the device manager into a more understandable format'.

(c) The **file manager** controls all the different files on the system. It controls file permissions, such as a user's ability to see or open a file, write to a file or delete it. It is therefore important for the security of the system. File management also helps organise and control files so that they are as user-friendly as possible'. It helps to protect the user from accidental mistakes too.

Worksheet 12.2

1 (a) **Utility systems** software perform specific tasks related to computer functions, resources, files and security. They help to configure the system, analyse how it is working and optimise it, improving its efficiency.

© Cambridge University Press

(b) (i) **Disk defragmentation** tools are used to rearrange the parts of files on the disk drive. When a file is saved to a disk, parts of the file might be saved in different areas of the disk. These tools try to move all the parts to the same area so that they can be accessed more quickly.

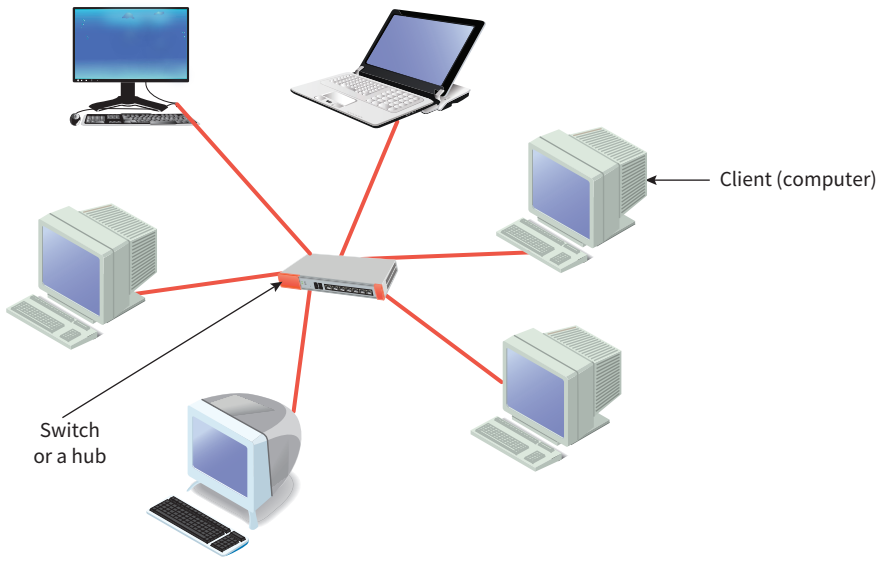
(ii) **File compression** software is used to make the files smaller so they take up less storage space and can be transmitted to other users more easily.

(c) (i) **Encryption software** uses an algorithm to encrypt (scramble) a file according to the key which is used; the key is needed to decrypt the file back to its original form. The file can therefore only be opened by authorised access..

(ii) **Firewalls** are either software or hardware devices that protect against unauthorised access to a network, and are primarily used to prevent unauthorised access from the Internet. They can be configured to prevent communications from entering the network and also to prevent programs and users from accessing the Internet from the network.

Worksheet 13.1

1 (a) Diagram to show computers connected to a single point.



Using a star topology, each computer is connected individually to a central point, which can be a file server or switch. The computers are individually connected by cable to a switch. The cable is connected to the network adapter in each device.

Required hardware: A network adapter for each computer, cabling, and a switch or a hub.

(b) **Advantages:**

- Easy and cheap to install as only access points are required.
- Very mobile. Users can access the network from anywhere on the site. They can move from room to room and remain connected.

Disadvantage: Any one from:

- Far lower bandwidth leading to lower download and upload speeds.
- Security is poor. Anyone within range can see the network and connect to it to use it. The access point must be secured with a security password and some form of encryption must be set up.

© Cambridge University Press

- The signal can be affected by walls and electronic equipment such as microwave ovens. It is also affected by distance from the access point and the number of connected computers.

Worksheet 13.2

1 The answer should contain the following points:

- The school/technical team are responsible for backing up/disaster recovery if the data is stored on hard discs connected to the school's servers.
- The storage provider is responsible for backup/disaster recovery if the school stores its data in the 'cloud'.
- The school is responsible for security of the files.
- The storage provider is responsible for security and 'cloud' storage is a target for hackers.
- The school's files are available without an internet connection if data is stored on hard discs connected to the school's servers.
- An internet connection is required to access the school's files if they are stored in the 'cloud'.
- Staff at the school can access their files anywhere by using mobile devices if the school's files are stored in the 'cloud'.
- The staff can access their files at school only if the data is stored on hard discs connected to the school's servers.
- The school will be responsible for maintaining the servers, replacing them when they age or the technology becomes outdated.
- The storage provider is responsible for updating the storage system in the 'cloud' and providing new technology as it evolves.
- The school will be responsible for increasing the storage capacity of their servers as demands increase.
- The storage provider will be responsible for increasing the storage capacity of their servers as demands increase.
- If confidential data is stored on site, then the school can ensure that the Data Protection Act is complied with.
- Storing data online may contravene the Data Protection Act.

Worksheet 13.3

1 (a) **Domain names** are used to identify IP addresses as they are more convenient to use and easier to remember than the four octets of binary numbers (e.g. mysite.org is easier to remember and enter than 192.100.321.003).

(b) **The domain name service (DNS)** returns the IP address when a client requests access using a domain name. When a browser requests access to a host using its domain name, the client computer contacts a DNS server. The DNS server contains a database of domain names which allows it to look up the domain name and returns the IP address. This is known as resolving the domain name.

© Cambridge University Press

2 When devices transmit data, the data are broken down into small pieces called packets. These are sent separately, and then joined up at the end so that the message is complete. This process is called packet switching.

- These packets are then sent onto the network using cables or microwaves as in a wireless network.
- Routers on the network inspect each packet and decide on the most efficient path for the packet to take on the next stage of its journey.
- To do this, each router has a configuration table containing information about which connections lead to particular groups of addresses.
- The routers can balance the load across the network on a millisecond-by-millisecond basis.
- If there is a problem with one part of the network while a message is being transferred, packets can be routed around the problem, ensuring the delivery of the entire message.
- The final router can direct the packet to the correct recipient.

Worksheet 13.4

1 (a) **TCP** (Transmission Control Protocol)/**IP** (Internet protocol).

(b) Transport layer

Internet layer or Network layer

Data link layer or Network access layer.

(c) (i) and (ii) any two from:

- **FTP (File Transfer Protocol):** provides the rules that must be followed when files are being transmitted between computers.
- **HTTP (Hypertext Transfer Protocol**): the rules to be followed by a web server and a web browser when requesting and supplying information. HTTP is used for sending requests from a web client (a browser) to a web server and returning web content from the server back to the client.
- **HTTPS Secure HTTP:** ensures that communications between a host and client are secure by ensuring that all communication between them is encrypted.
- **SMTP (Simple Mail Transfer Protocol):** the protocol for sending email messages from client to server and then from server to server until it reaches its destination.
- **POP (Post Office Protocol):** used by a client to retrieve emails from a mail server. All of the emails are downloaded when there is a connection between client and server.
- **IMAP (Internet Message Access Protocol):** unlike POP, the messages do not have to be downloaded. They can be read and stored on the message server. This is better for users with many different devices as they can be read from each one rather than being downloaded to just one.

(d) Any two from:

- It simplifies the overall model by dividing it into functional parts.
- Each layer is specialised to perform a particular function.

© Cambridge University Press

- The different layers can be combined in different ways, as required.
- One layer can be developed or changed without affecting the other layers.
- It makes it easier to identify and correct networking errors and problems.
- It provides a universal standard for hardware and software manufacturers to follow, so that they will be able to communicate with each other.

Worksheet 14.1

1 Answer to include:

- Users often pose the greatest threats to networks, either through their direct actions or by allowing criminals access to the networks illegally.
- Users can pose a threat to network security by using their own portable devices. These devices pose a threat because they can introduce malware to the network or remove data.
- If data is removed illegally, the company may be sued if this data is covered by the Data Protection Act.
- Network policies should cover the use of removable media such as USB flash drives, smartphones, CDs, DVDs, MP3 players and digital cameras.
- Network policies may state that only devices provided by the company can be used.
- Network policies should cover user authentication and the use of passwords.
- Network policies should ensure that users do not use passwords which are easy to remember and are based on personal details such as birth dates and the names of family members.
- All user passwords are at least eight characters long and contain numbers, letters (upper and lower case) and non-alphanumeric characters such as exclamation marks.
- Passwords should be changed regularly and old ones should never be reused.
- Users may be susceptible to social engineering (tricking people into divulging secret information or doing things that they would not otherwise do).
- Social engineering techniques include:
  - o 'blagging', where a criminal uses a voice call or email to try to get a user to divulge information
  - o 'phishing', which is the use of fraudulent emails
  - o 'shouldering', where a user can be watched or filmed entering user names and passwords.
- Network policies should ensure that users do not divulge any details.
- Users should be given information about, and training on how to deal with, suspicious emails.
- Users can be a risk to the networks they use and they should be given security training, reinforced by strict network policies.

© Cambridge University Press

Worksheet 14.2

1 (a) **Denial of service** attacks flood a network or website with useless network communications, such as repeated login requests, which prevent legitimate users from gaining access to the network or website. They are caused by hackers taking over thousands of computers which they then use in the attack.

(b) Many websites use databases to store users' details such as names, addresses, credit card details, etc. SQL (structured query language) is used legitimately to analyse this data and carry out business activities. In **SQL injection**, criminals input specially created commands instead of a username or password. These commands gain access to the database so that the criminals have access to users' data.

(c) During **data interception** attacks, criminals use software called packet analysers or packet sniffers to intercept network packets. The packets are analysed and decoded. This allows criminals to steal sensitive data such as logins, passwords, credit card numbers and PINs.

Worksheet 14.3

1 (a) **Encryption** is the scrambling of data into a form that cannot be understood by unauthorised recipients. The encrypted data must be decrypted back to its original form.
A common method is the use of a 'public' and a 'private' key. The public key is freely available to anyone, but the private key is only known to the owner. Messages encrypted by a particular public key can only be decrypted with the corresponding private key.

(b) **Penetration testing** is the testing of a computer system, network or web application to find vulnerabilities that an attacker could exploit. The test then indicates how those vulnerabilities could be exploited to demonstrate the security risks.
The main objective is to determine security weaknesses. It can also be used to test an organisation's security policy, the security awareness of the users, and the organisation's ability to identify and respond to security incidents.

There are two types:

- White-box penetration testing, which simulates hacking by 'insiders', people who have full knowledge of the network. It is also called 'full disclosure' testing as the testers are given details of items such as IP addresses, source code, network protocols and even login names and passwords.

- Black-box penetration testing, which is also called 'blind testing' because testers are given very little or no information. The testers must find their own way into the network without any knowledge or normal means of access. Black-box testing is more realistic to everyday penetration attacks.

(c) Firewalls are either software or hardware devices that protect against unauthorised access to a network, and are primarily used to prevent unauthorised access from the Internet.

They can be configured to prevent communications from entering the network and also to prevent programs and users from accessing the Internet from the network. A firewall, for example, can inspect the incoming packets and reject those that are not from a trusted IP address list or block communication to certain external IP addresses.

Worksheet 15.1

1 Answer to include:

Energy

- Manufacturing involves energy-intensive mining and processing of minerals.

- The use of devices involves the energy used by the devices themselves, but also by data centres. These data centres generate heat, so energy is needed to keep them cool.

- Much of the energy used comes from non-renewable sources such as gas and coal.

© Cambridge University Press

- Computer science is used in efficient energy production.
- Computer software is used to design, model and test efficient devices to produce electricity from wind, wave and solar power.
- Energy use can be reduced using smart technologies, such as light- sensitive switches that turn off lights when they are not needed.
- Efficient transport planning using computer modelling and analysis can reduce fuel use.

Sustainability

- Digital devices use many different chemical elements. Some of these are rare and will be in short supply as they are used up.
- It is difficult to recycle devices to reuse these elements.

Waste

- Electronic devices are difficult to recycle and are often disposed of in landfill sites as e-waste.
- Landfill sites take up areas of land that could be used for other purposes.
- Toxic substances such as lead, mercury and cobalt can get into the soil and the water supply from the landfill sites and so cause health problems.
- Often old computing devices are dumped illegally in third world countries where toxic fumes (caused by reprocessing and leakage) cause health problems and death.

Data analysis

- Computer science technology can be used to monitor environmental factors by transmitting and analysing data.
- This data can be shared by scientists around the world who can collaborate to find solutions to problems.
- Computers can be used to develop models to forecast environmental behaviour and identify options for action.

Worksheet 15.2

1 (a) Any three from:

- Data will be kept securely.
- Data must be accurate and up-to-date.
- Data can only be used for the purpose for which it was collected.
- Only data that is actually needed should be held.
- Data must not be held longer than it is needed for.
- Data will be used in accordance with the rights of the data subjects.

© Cambridge University Press

(b) Any three from:

- A right of access to a copy of the information contained in their personal data.
- A right to object to useage that is likely to cause or is causing damage or distress.
- A right to have inaccurate personal data rectified, blocked, erased or destroyed.
- A right to claim compensation for damages caused by a breach of the Confidentiality/ Protection Act.
- A right to prevent usage for direct marketing.

Worksheet 15.3

1 (a) **Open source software** is freely available on the Internet, free to use and is constantly being upgraded by users. Under the terms of the licence, they are obliged to charge no fee if they pass it on to other users.

The four key freedoms of the open source movement are:

1. The freedom to run the program for any purpose.
2. The freedom to study how the program works and change it.
3. The freedom to redistribute copies.
4. The freedom to distribute copies of modified versions to others.

(b) Should include the following points:

Benefits:

- It is free to use.
- Source code can be viewed and modified.
- It is constantly being upgraded and modified by a group of enthusiasts.
- Free advice and help.

Drawbacks:

- It may need specialist help to download and install the software.
- It may not have been tested as thoroughly as proprietary software, which is developed professionally and carefully tested.
- There will be books, magazine articles and online tutorials available for proprietary software.

© Cambridge University Press

Acknowledgements

The authors and publishers acknowledge the following sources of copyright material and are grateful for the permissions granted. While every effort has been made, it has not always been possible to identify the sources of all the material used, or to trace all copyright holders. If any omissions are brought to our notice, we will be happy to include the appropriate acknowledgements on reprinting.

Screenshots in Chapter 10 used with permission from Stephen Chen, Associate Professor, School of Information Technology.

© Cambridge University Press