1 Algorithms

Learning outcomes

By the end of this chapter you 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 charts and pseudocode
- analyse, assess and compare different algorithms
- create, name and use suitable variables
- use arithmetic, relational and Boolean operators
- use conditional statements.

Challenge: create an algorithm to help a taxi company calculate its fares

- By the end of this chapter, you should have a thorough knowledge of how algorithms can be used to solve complex problems and how they can be displayed using flow charts and pseudocode.
- Your challenge is to use this knowledge to help a taxi company calculate its fares.

Why algorithms?

Algorithms run our world! In every area algorithms are used to decide what action should be taken in a particular circumstance and as computers can consider all the possibilities far more quickly than a human brain, they are becoming more important to the running of the world. Here are just a few examples.

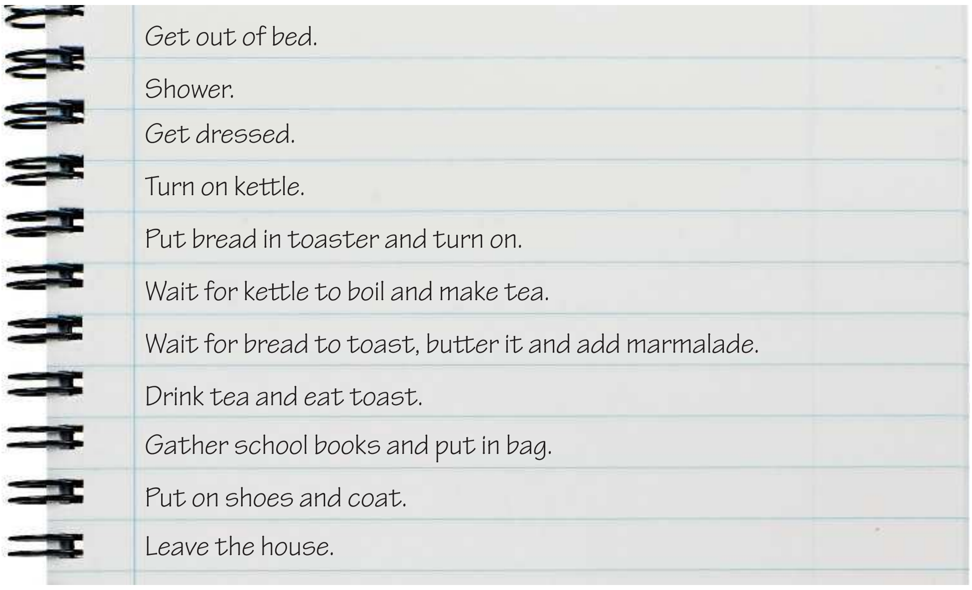
- In a game of chess, when each player has made 3 moves, there are over 9 million possible moves available; after 4 moves there are over 288 billion possible moves. Computers have the ability to consider all these possible moves far more quickly than humans. That is why no chess grandmaster has beaten a top computer chess algorithm since 2005.
- Algorithms are used by financial organisations to trade shares on the stock market. A computer following an algorithm can decide which deal to make far more quickly than a human and a split second difference can be worth millions of pounds.
- Closely guarded algorithms are used for Internet searches to make them quicker and the results more relevant to the user. They will even auto-complete the search terms based on previous searches.

Algorithms are used to control automatic-pilot systems in airplanes. You have probably been piloted by an algorithm!

What is an algorithm?

An algorithm is a step-by-step procedure for solving problems. It is something that can be followed by humans and computers.

We use algorithms to carry out everyday tasks, often without thinking about them. For example, an algorithm to solve the problem of getting ready for school might be:



Get out of bed.
Shower.
Get dressed.
Turn on kettle.
Put bread in toaster and turn on.
Wait for kettle to boil and make tea.
Wait for bread to toast, butter it and add marmalade.
Drink tea and eat toast.
Gather school books and put in bag.
Put on shoes and coat.
Leave the house.

Watch out

In an algorithm, the order in which the tasks are carried out is very important to its success or failure. For example, this algorithm would not be very successful if 'shower' was placed after 'get dressed'. The sequence is very important.

The algorithm shows the sequence of tasks. Different people will design different algorithms, as they will do things in a different order, meaning there can be many solutions to the same problem. Some of these tasks could also be further divided into sub-tasks as they may be made up of smaller steps.

Key terms

sequence: the order in which tasks are to be carried out

sub-tasks: small steps making up a larger task

For example, 'showering' could involve many different steps: turning on the shower, setting the correct temperature etc. If all the possible sub-tasks were included, the complete algorithm would get very large and complicated.

ACTIVITY 1.1

Create an algorithm for someone who has never made a cup of tea before to follow in order to make one successfully. Compare it with other members of your group and note any differences in sequence and sub-tasks.

Complete the Cambridge Computing Online activity
www.cambridge.org/links/kose4001

Download Worksheet 1.1 from Cambridge Elevate

The getting to school example might seem pretty easy. Like a typical algorithm, it is just a list of steps. However, here is part of another algorithm, which is part of a recipe to make a perfect meringue.

1. Tip 4 large egg whites into a large clean mixing bowl (not plastic).	SEQUENCE
2. Beat them on medium speed with an electric hand whisk.	SEQUENCE
3. Keep beating until the mixture resembles a fluffy cloud and stands up in stiff peaks when the blades are lifted.	ITERATION SELECTION
4. Now turn the speed up and start to add 115g caster sugar, a dessert spoonful at a time, until there is none left.	ITERATION SELECTION
5. Continue beating for 3–4 seconds between each addition.	SEQUENCE

In addition to *sequence*, this algorithm has two new elements: iteration and selection.

Iteration means doing things over and over again. The cook has to beat the mixture and then stop and ask themselves if it resembles a fluffy cloud. If it doesn't they have to beat again, check again, beat again, check again etc. until they are convinced they have made a fluffy cloud. There is also repetition when adding the sugar. It has to be added a spoonful at a time until there is none left.

Selection means making decisions. In other words, as well as doing things over and over again, they have to make a decision. Does it resemble a fluffy cloud?

Key terms

iteration: when a task is repeated until there is a required outcome

selection: a question is asked, and depending on the answer, the program takes one of two courses of action

Complete Interactive Activity 1a on Cambridge Elevate

ACTIVITY 1.2

Using *sequence*, *selection* and *iteration*, write an algorithm that a person (who has never done this before) could follow in order to successfully prepare a bath with the water at the correct temperature. Annotate the algorithm to indicate sequence, iteration and selection.

You could set it out as shown, where the first four tasks have been done for you.

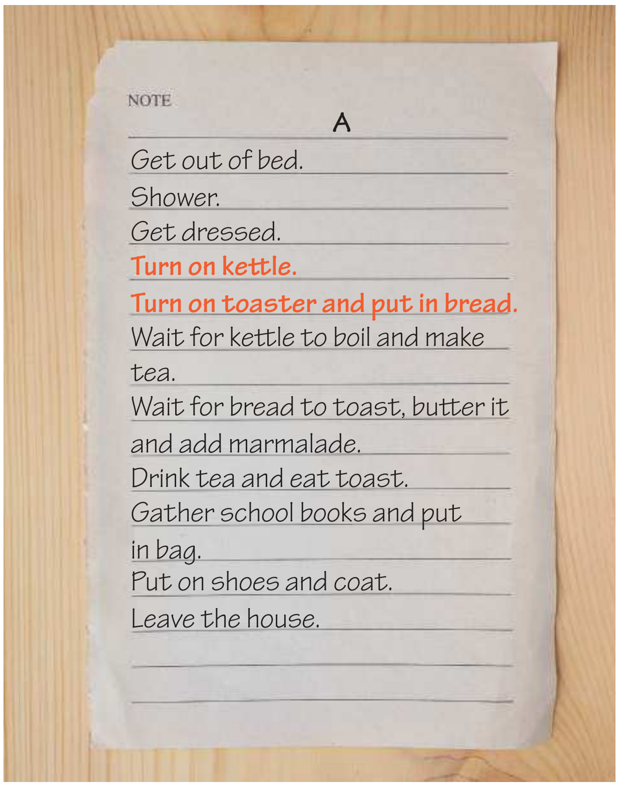
`Put plug in the bath.`	Sequence
`Turn on hot tap.`	Sequence
`Is the water at the correct temperature?`	Selection
`Turn cold tap until water is at the correct temperature.`	Iteration

What makes a successful algorithm?

The two most important criteria are:

- *Correctness*: it successfully solves the problem.
- *Efficiency*: it solves the problem in the least possible time.

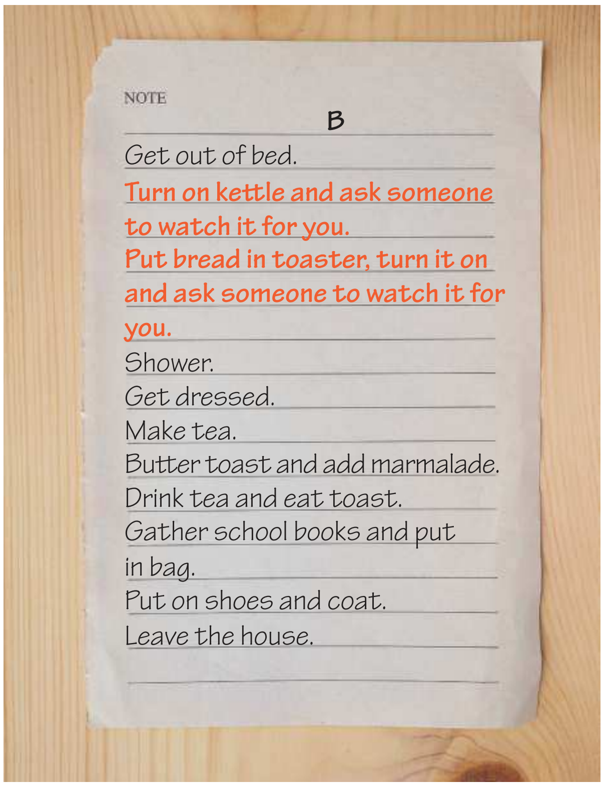
List A is the 'getting up' algorithm we looked at earlier. **List B** is similar but with the sequence slightly altered.



NOTE

A

Get out of bed.
Shower.
Get dressed.
Turn on kettle.
Turn on toaster and put in bread.
Wait for kettle to boil and make tea.
Wait for bread to toast, butter it and add marmalade.
Drink tea and eat toast.
Gather school books and put in bag.
Put on shoes and coat.
Leave the house.



NOTE

B

Get out of bed.
Turn on kettle and ask someone to watch it for you.
Put bread in toaster, turn it on and ask someone to watch it for you.
Shower.
Get dressed.
Make tea.
Butter toast and add marmalade.
Drink tea and eat toast.
Gather school books and put in bag.
Put on shoes and coat.
Leave the house.

List B is more efficient as it could be implemented in less time. The kettle and the toaster are turned on before taking a shower and so the water will boil while the person is showering. There will be no waiting time.

Watch out

Although the algorithm is now more efficient, it would not have been safe to leave the kettle and toaster unattended. Computer scientists must consider health and safety issues when designing real-life solutions.

An algorithm for a computer

Now let's look at a simple algorithm that we could create for a computer to follow, instead of a human. Computers are ideal for obeying orders and carrying out actions over and over again; in fact, that is their main function.

It is important for the temperature in a shopping mall to be kept at a set value. It will keep the shoppers comfortable and it will help to prevent condensation on glass shop windows and slippery floor surfaces. Here is an algorithm intended to be used to control the temperature in a shopping mall and maintain a temperature of 20 °C.

1. Check the temperature.
2. If the temperature is greater than 20 °C then turn off heaters and open the ventilators.

3. If the temperature is less than 20 °C then turn on heaters and close the ventilators.

4. Go back to instruction 1.

This is a simple algorithm but it includes the basic building blocks:

- *Sequence*: there is a list of instructions in the correct order.
- *Selection*: the 'if' statements in instructions 2 and 3 allow a decision to be made and action taken.
- *Iteration*: instruction 4 instructs the computer to go back to instruction 1 and so the sequence will run over and over again indefinitely.

Remember

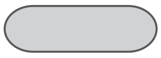
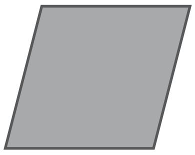
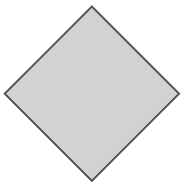
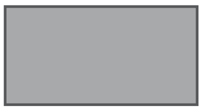
1. An algorithm is a step-by-step procedure for solving a problem in a finite number of steps.
2. The basic building blocks of algorithms are sequence, selection and iteration.
3. The criteria for a successful algorithm are correctness and efficiency.
4. An algorithm must be translated into a programming language before it can be executed by a computer.

Flow diagrams

Flow diagrams can be used to represent algorithms visually.

They use symbols connected by arrows to show the flow of the algorithm.

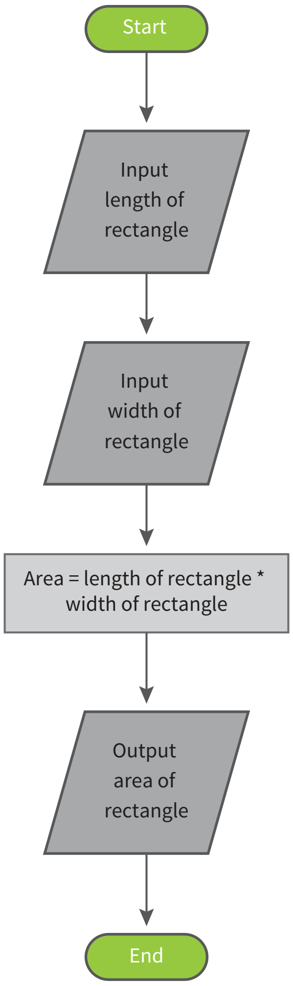
The symbols used are:

This represents the start or end point of the flow diagram. You always start your flow diagram with this symbol and use it to finish the flow diagram.	
You use this to represent data input or data output. For example it could be a number or name entered by a user.	
You use this where a decision has to be made. It is also called selection. It will contain a question, for example 'Is the temperature greater than 20 °C?' If it is then an arrow will point to a task to be carried out and if it isn't then an arrow will point to a different action.	
You use this to represent a process that must be carried out by the algorithm. In this example it is used as a result of one of the questions which has been asked, e.g. 'Turn on the heaters' or 'Turn off the heaters'.	

Watch the flow diagrams animation on Cambridge Elevate

WORKED EXAMPLE

Here is the flow diagram of an algorithm to calculate the area of a rectangle:


Start
Input length of rectangle
Input width of rectangle
Area = length of rectangle * width of rectangle
Output area of rectangle
End

This start/stop symbol is used at the beginning and end of an algorithm.

This is input. The user is being asked to enter the length of the rectangle.

This is input. The user is being asked to enter the width of the rectangle.

This is a process to be carried out by the computer. It calculates the area of the rectangle.

This is output. The area of the rectangle is displayed (output) for the user.

This start/stop symbol is used at the beginning and end of an algorithm.

There are two inputs of the dimensions of the rectangle, a process to calculate the area and an output of the area.

In this example there is just *sequence*: a list of tasks to be performed.

Complete Interactive Activity 1b on Cambridge Elevate

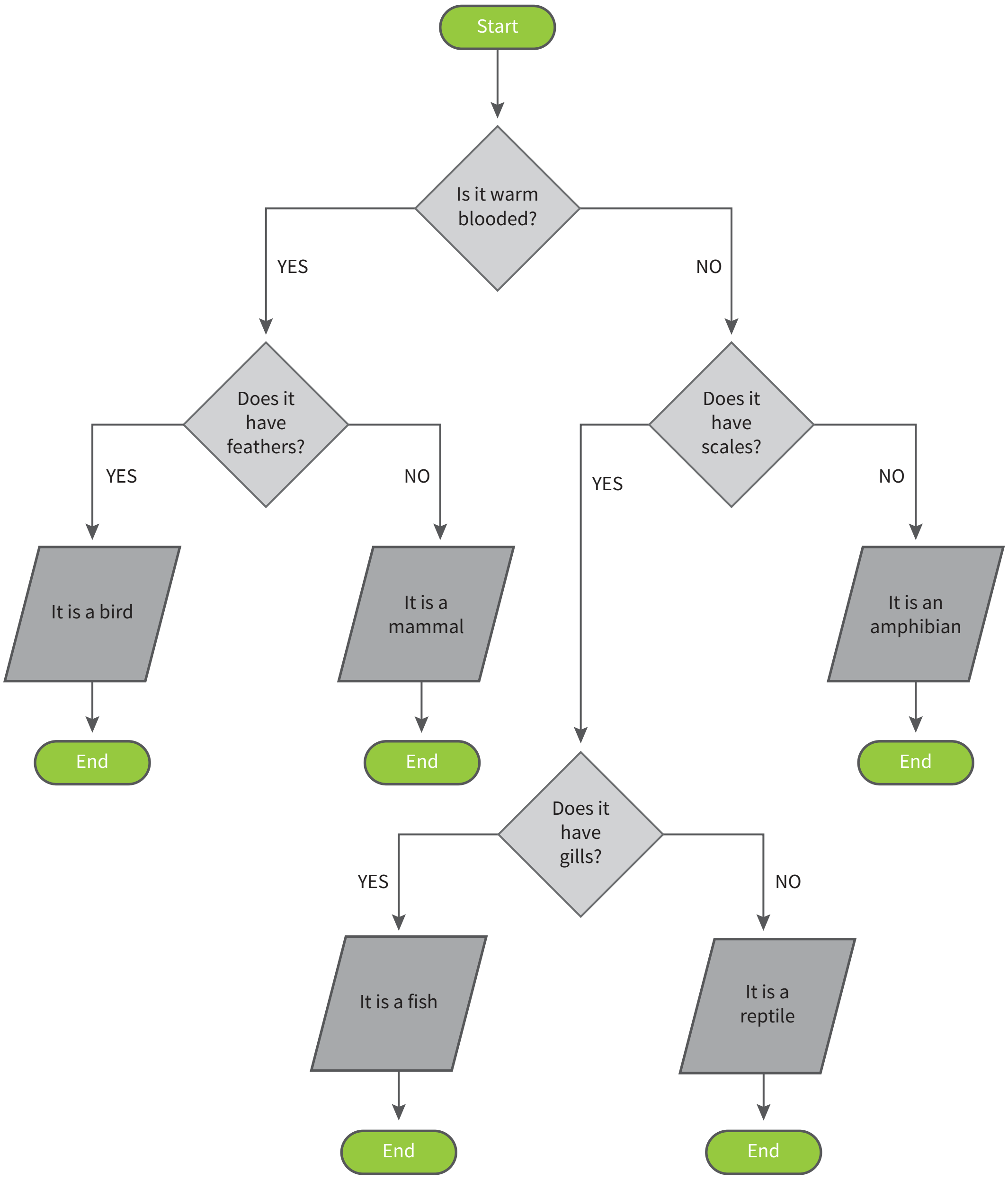
ACTIVITY 1.3

At the end of each day an ice cream seller calculates how much money he has collected. Assuming that the ice creams all cost the same amount, draw a flow diagram of an algorithm that would output the total amount collected during the day.

Cambridge University Press
978-1-316-50403-1 — GCSE Computer Science for OCR Student Book
David Waller , Course consultant Ann Weidmann
Excerpt
More Information

WORKED EXAMPLE

Here is a flow diagram of an algorithm to identify a vertebrate animal. It includes *sequence* and *selection*.



Complete Interactive Activity 1c on Cambridge Elevate

ACTIVITY 1.4

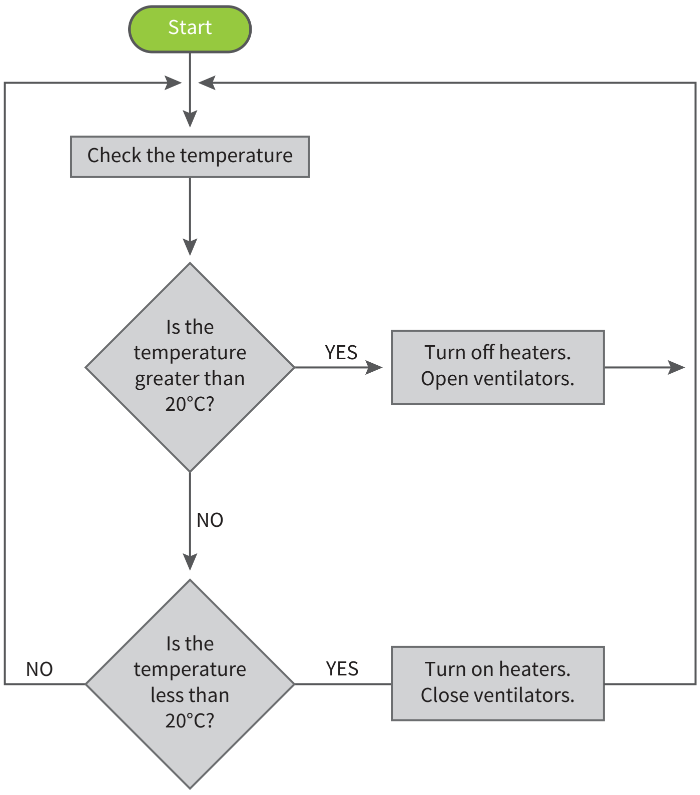
A teacher is marking his students' test papers on a computer. If they achieve over 50% he would like the message 'Very well done' displayed. If they achieve over 90%, they should also receive a second message stating 'This is an excellent result'. If they score 50% or lower, the message will be 'You must try harder next time'.

Draw a flow diagram of an algorithm that would output these messages.

CAMBRIDGE
Cambridge University Press
978-1-316-50403-1 — GCSE Computer Science for OCR Student Book
David Waller , Course consultant Ann Weidmann
Excerpt
More Information

WORKED EXAMPLE

A flow diagram to represent the algorithm to control the temperature of the shopping mall that we mentioned earlier would look like this. It contains *sequence*, *selection* and *iteration*.


Start
Check the temperature
Is the temperature greater than 20°C?
YES
Turn off heaters. Open ventilators.
NO
Is the temperature less than 20°C?
NO
YES
Turn on heaters. Close ventilators.

There are three **processes**: one to check the temperature in the mall and two to either turn off the heaters and open the ventilators, or turn them on and close the ventilators.

There are two **decisions**: is the temperature greater or is it less than 20 °C? Both are needed as the temperature could in fact be equal to 20 °C.

There are only two possible answers for each decision question: *yes* or *no*, and the arrows show the relevant action to be taken depending on the answer.

Iteration is shown in the flow diagram as the arrows always lead the flow back to the first process. So the algorithm will repeat over and over again indefinitely.

As the algorithm repeats forever, no end symbol is required.

Key terms

decision: when a question is asked (as in *selection*) the answer will lead to one or more different alternative actions

process: action that is taken, sometimes as a result of a decision, in order to achieve a desired outcome

ACTIVITY 1.5

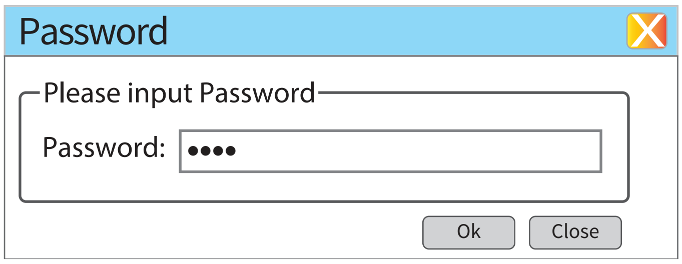
Draw a flow diagram to illustrate an algorithm for making a cup of tea.

It should include sequence, selection and iteration.

(**Hint**: Look back at your answers for Activity 1.1.)

Input and output

In some of the previous examples, user input was required and information was output to the user.


Password

Please input Password

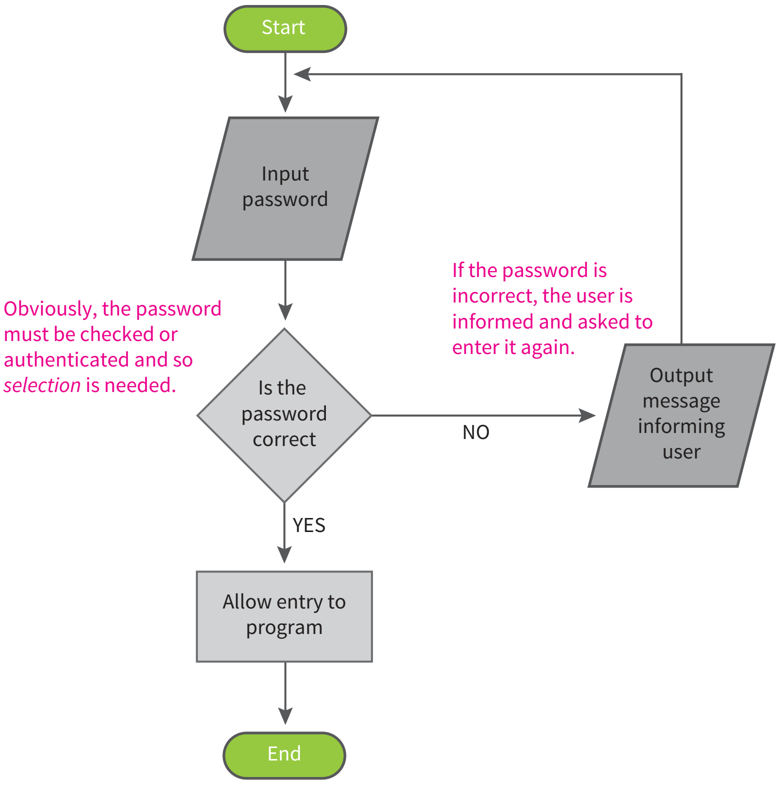
Password: ●●●●

A common request for user input is to enter a password.

WORKED EXAMPLE

Here is a flow diagram of an algorithm to authenticate a user's password.

When a user enters a password it has to be confirmed that it is the same as the one stored; it has to be authenticated.



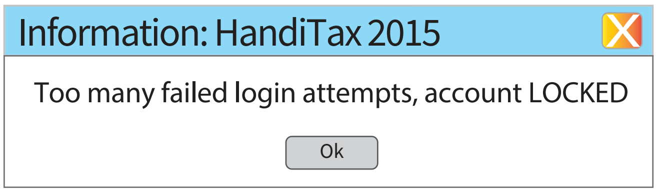
Obviously, the password must be checked or authenticated and so *selection* is needed.

If the password is incorrect, the user is informed and asked to enter it again.

In this flow diagram there is *iteration*. If the password is incorrect then, in this particular algorithm the user is asked to enter it again, and again, and again, forever or until it is correct.

Download Worksheet 1.2 from Cambridge Elevate

Now assume that the user is given three attempts and then the account is locked.


Information: HandiTax 2015

Too many failed login attempts, account LOCKED

Key term

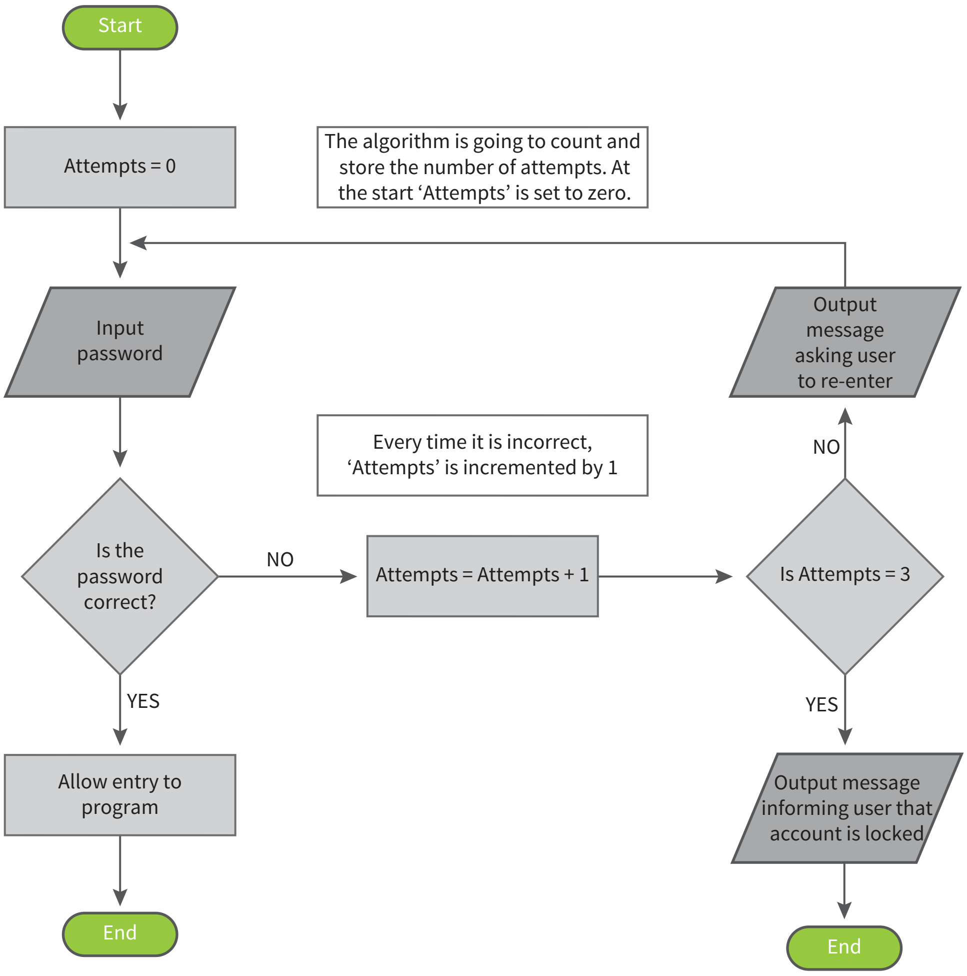
authenticate: confirm that a user's password has been entered correctly and matches the password stored on the system

Cambridge University Press
978-1-316-50403-1 — GCSE Computer Science for OCR Student Book
David Waller , Course consultant Ann Weidmann
Excerpt
More Information

WORKED EXAMPLE

Here is a flow diagram to authenticate a password and lock the account after three incorrect attempts.

The algorithm will have to keep a count of the number of attempts that have been made.


Start
Attempts = 0
The algorithm is going to count and store the number of attempts. At the start 'Attempts' is set to zero.
Input password
Output message asking user to re-enter
Every time it is incorrect, 'Attempts' is incremented by 1
NO
Is the password correct?
NO
Attempts = Attempts + 1
Is Attempts = 3
YES
YES
Allow entry to program
Output message informing user that account is locked
End
End

In this algorithm we have used a container called 'Attempts' to keep a count of the number of attempts that have been made.

When incorrect attempts are made, the value of 'Attempts' changes. It doesn't keep the same value throughout the algorithm; it can change because it is a variable.

At the first attempt it is changed to 1, on the second attempt to 2 and to 3 on the third attempt.

If three attempts are made then the container 'Attempts' equals 3 and if there is still no correct password then the account is locked.

Containers like 'Attempts' are used in algorithms to store values that can change as the algorithm is running. As the values they contain can change, they are called variables.

Key term

variable: a container which is used to store values such as the number of attempts

Download Worksheet 1.3 from Cambridge Elevate