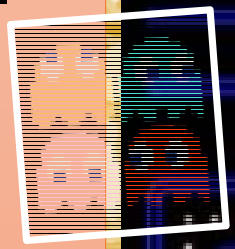
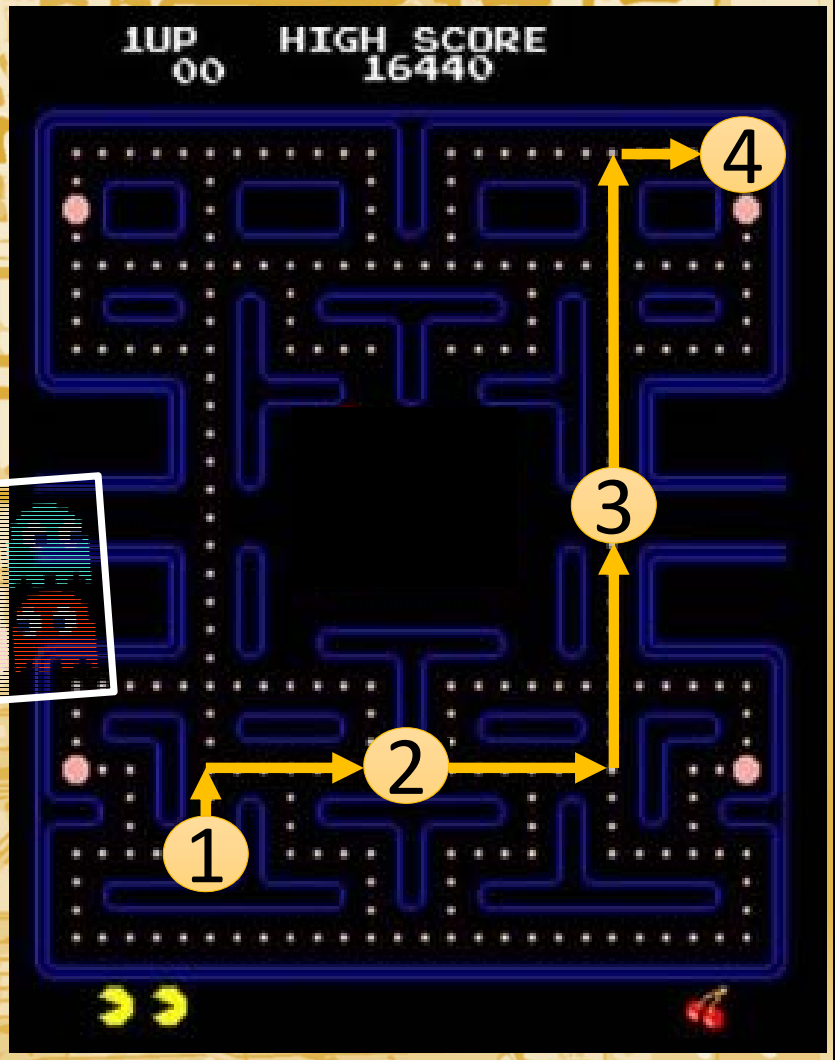
Summer Task #1

When people played the classic Pacman game, the ghosts were given names and were said to have *personality*; this was because each ghost was programmed to respond differently to the player's movements. On the image to the left you can see the players movement, going from location 1 → 2 → 3 → 4.
Using each ghosts' algorithm you should be able to predict *roughly* where each ghost will be when the player is at each location.

Task:

- Research each ghost's algorithm. Google: **Pacman Ghost Behaviour**
- Find a simple Pacman board similar to the one to the right
- Mark out 4 locations similar to the diagram
- For each location provide an approximate location for each of the four ghosts – you obviously can't know for sure but you should be able to make educated guesses. *For position 1 feel free to set up the ghost's location to make it easier to track for the following locations.* ☺
- Briefly justify your placements for the ghosts basing it on the ghost's algorithm



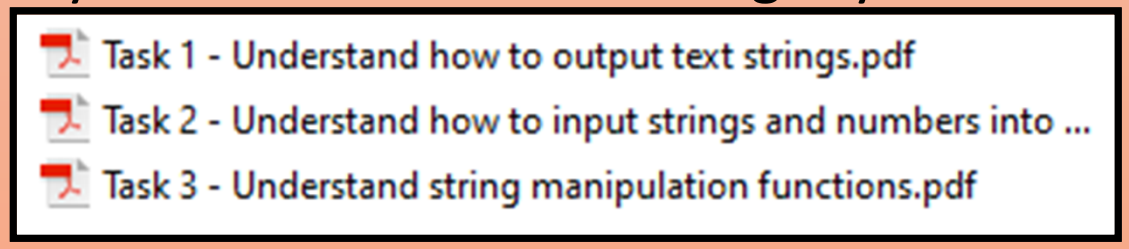


Summer Task #2 for students new to CS

Install Python

1. See **Getting started with Python.pdf**

1. Try the three task sheets to get you started with the basics

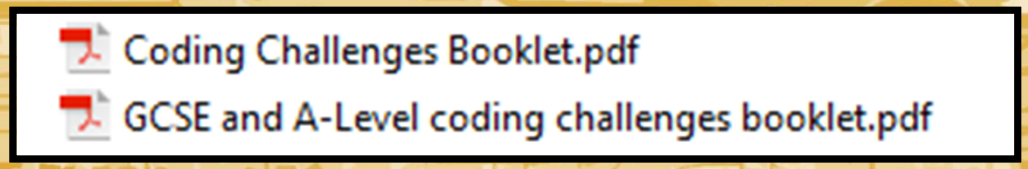


2. Evidence your work by taking screenshots of your programs and pasting on Word or PowerPoint or any software of your choice.

Summer Task #2 for existing CS students

Attempt **three** programs from the **GCSE Code Challenge** booklet. Take your time and thoroughly understand the program and what your solution will need e.g. variables, variables for holding temporary values, inputs, outputs etc.

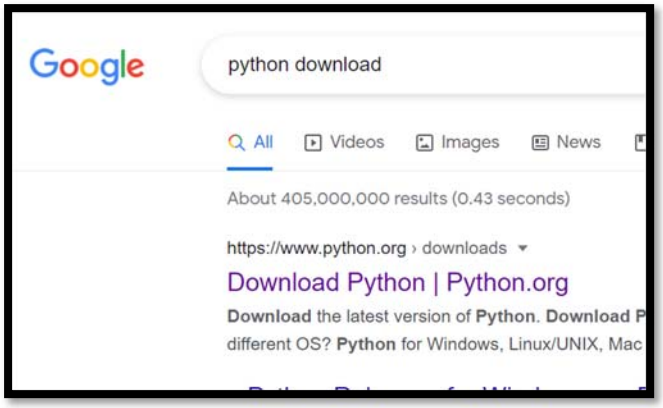
*Bonus challenge, pick another one from the **GCSE and A-Level coding challenges booklet** collection.* ☺



Getting started in Python

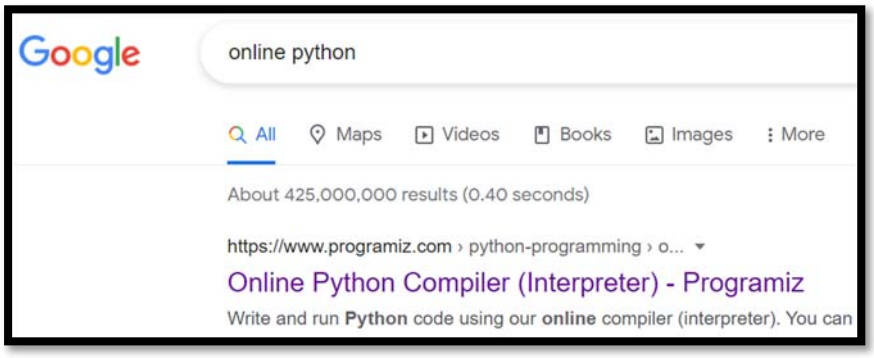
INSTALLATION

Start by downloading Python by Googling 'Python Download'



Then click on the download button or the link to the version for your operating system and then install. *It doesn't matter if the version number is different as long as it is Python 3 something.*

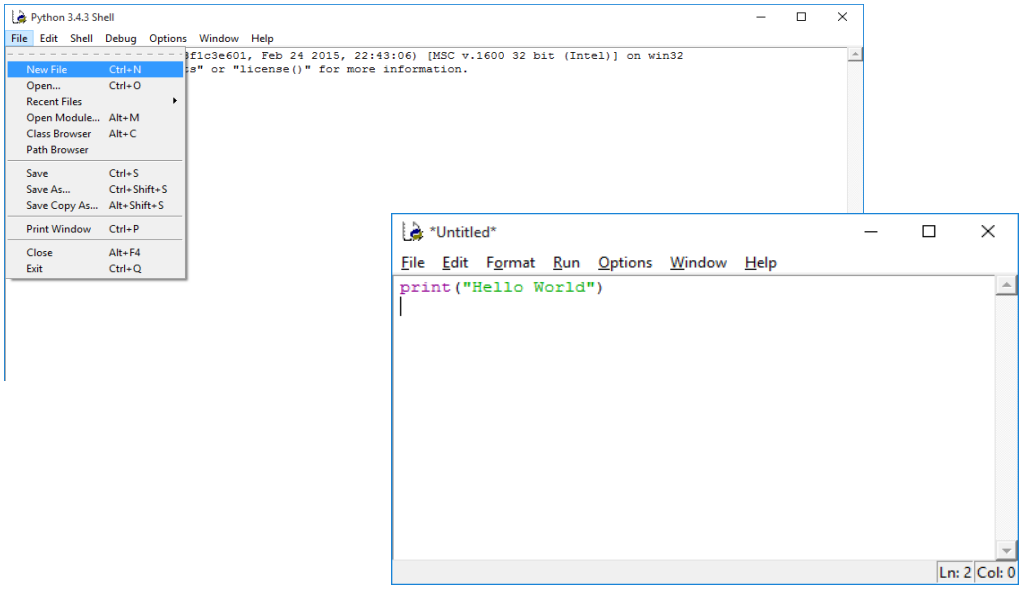
For those with a Chrome book or similar device you can use an online Python.



How to use

People using an online Python can skip to #4 because the 'Editor' window is usually already open along with the 'Shell'.

1. Open IDLE (Python).
2. Choose File… New File…



3. A new window will open into which you write your code.
4. The program must be saved before it will run.
5. To run your program press F5.
6. When you save your program, make sure to add .py extension to the filename. This will not only make it easier to find your programs, but will also ensure the text formatting is not lost in the editor.

Objective 1:

Understand how to output text strings

In this objective you learn how to output text to the screen.

Often, we need a computer to process (do something with) text. If we want a computer to store *text data*, the type of data required to store text is known as a '***string***'. To hold a whole number, the data type is known as '***integer***'. To hold a number with a decimal point, the data type is known as '***float***'. There are a couple of other **basic data types** but this will do for now. 😊

Tasks

1. Try entering the following commands and see what happens:

```
print("Hello World")
```

2. Try entering the text without the speech marks and see what happens:

```
print(Hello World)
```

Note the difference. One works and one doesn't! If you want to output text to the screen it must be included in speech marks. Text is called a string in programming and must be enclosed in double quotes. Python is a case sensitive language, the commands, such as 'print' must be entered in lowercase.

3. Try entering the following commands and see what happens:

```
print("Hello World","this is my first program.")
```

Note how a comma joins strings together. This is called concatenation. In Python a comma adds a space between the strings. If you don't want this, use a + instead of a comma.

4. Try this sequence of instructions:

```
#Start of message
print("Hello World","this is my first program.")
#Blank line
print()
#End of message
print("I am learning to code...")
print("...and it is fun")
```

Note the hash symbol enables you to insert a comment in the code. The computer ignores the comments, but they help to introduce the program and allow you to leave notes in the code to make it easier to understand later.

5. Change the program so it outputs this instead:

```
Computers only do exactly as they are told…
…so you need the code to be correct!

If you make any mistake with the commands, it won't work
```

Objective 1: Key learning points
Understand how to output text strings

- Text is known as a **string** in programming and must be enclosed in double quotes.
- Strings can be joined together using a comma or plus symbol. This is known as **concatenation**.
- When a program is run it is called **executing** the program.
- A set of instructions that execute one after another is known as a **sequence**.
- Comments can be inserted into the code using a hash symbol. These are helpful to make notes within the program so that the code can be understood again at a later date or by another programmer.

Objective 1: Key words

print()

Example code: `print(x)`

Purpose: to output x to the screen followed by a carriage return.

Carriage return means new line. The print command always assumes you want to add a new line after.

To disable the carriage return, use: `print(x,end='')`

Objective 2:

Understand how to input strings and numbers into variables

In this objective you learn how to get input from the keyboard to use in your program.

Tasks

1. Try entering the following commands and see what happens:

```
#Inputting strings in Python
print("Hello")
name_entered = input("What is your name? ")
print("Thank you",name_entered)
```

2. Try entering the following commands and see what happens:

```
year = int(input("What year is it please? "))
print("Ah, it is",year,"thank you.")
```

3. Change the program so it asks you for your name and your age, outputting for example:

```
Thank you Dave. You have registered an age of 15.
```

Objective 2: Key learning points

How to input strings and numbers into variables

- Data is input by a user into a **variable**.
- Variables have a data type: string, integer or float as examples, indicating how much memory they will use and the type of data they will store.
- Python does not require variables to be declared before they can be used.

Objective 2: Key words

input

Example code: `x = input("Enter your name:")`

Purpose: to store text input at the keyboard into a variable, x which can be used later in the program without inputting again.

int

Example code: `x = int(x)`

Purpose: convert variable x to an integer. Most useful to convert a string input to a number because the character "5" is not the same as the number 5 to a computer.

Combining input and int enables the input of a number. E.g.

```
x = int(input("Enter your age:"))
```

float

Example code: `x = float(x)`

Purpose: convert variable x to a floating point (decimal) number. Most useful to convert a string input to a number with decimal places because the characters "5.5" are not the same as the number 5.5 to a computer.

Objective 3:

Understand string manipulation functions

In this objective you learn how to extract and change data that has been input.

Tasks

1. Try entering the following commands and see what happens:

```
#Working with strings
forename=input("Enter your surname: ")
forename_uppercase=forename.upper()
print("Your name in capital letters is:",forename_uppercase)
```

2. Change the program to ask for an email address, outputting the string in lowercase using `.lower()` instead of `.upper()`.

3. Try entering the following commands and see what happens:

```
#Len returns the number of characters in a string
surname = input("Enter your surname: ")
length_name = len(surname)
print("There are",length_name,"letters in your name.")
```

4. Try entering the following commands and see what happens:

```
#[:?] returns a number of characters to the left of a string
sentence = "I saw a wolf in the forest. A lonely wolf."
characters = sentence[:5]
print(characters)
```

5. Change the program to output the last 12 characters in the sentence using `[-12:]` instead of `[:5]`.

6. Try entering the following commands and see what happens:

```
#[start:end] returns a number of characters in the middle of a string
sentence = "I saw a wolf in the forest. A lonely wolf."
characters = sentence[20:26]
print(characters)
```

7. Try entering the following commands and see what happens:

```
#find returns the location of one string inside another
sentence = "I saw a wolf in the forest. A lonely wolf."
print(sentence)
word = input("Enter the word to find: ")
position = sentence.find(word)
print("The word",word,"is at character",position)
```

Objective 3: Key learning points

String manipulation functions

- Strings can be manipulated using built in functions and methods to extract characters from the left, right or middle of a string.
- You can find if one string exists inside another string.
- A built in function takes data to use in parenthesis (brackets), called a **parameter** and returns a result. E.g. `int(parameter)`
- A method (signified with a dot) applies an operation to itself. E.g. `"Hello".upper()`

Note, it is possible for functions and methods to return their value to another command rather than a variable. For example: `print("Hello World".find("World"))` works because the resultant value from find becomes the parameter for print.

It is common in programming to use as few variables as necessary in order to conserve memory. This makes the program more efficient, but often more difficult to understand.

Objective 3: Key words

.upper()

Example code: `x = y.upper()`

Purpose: To turn a string into uppercase.

x is the name of the variable to return the result to. y is the original variable.

.lower()

Example code: `x = y.lower()`

Purpose: To turn a string into lowercase.

len()

Example code: `x = Len(y)`

Purpose: To return the number of characters in a string.

x becomes the number of characters in y.

str()

Example code: `x = Str(4)`

Purpose: To return a string from a number.

If you wanted to output an integer or float variable with a string, you need to convert the number to a string first. E.g. to output 4 * 3 =

This would not work: question = number1 + "*" + number2 + "="

Instead, you need to cast the numbers to strings like this:
question = str(number1) + "*" + str(number2) + "="

[:?]

Example code: `x = y[:z]`

Purpose: To return characters to the left of a string.

x becomes the characters. y is the string to extract characters from. z is the number of characters to extract from the left.

[-?:]

Example code: `x = y[-z:]`

Purpose: To return characters to the right of a string.

x becomes the characters. y is the string to extract characters from. z is the number of characters to extract from the right.

[?:?]

Example code: `x = y[w:z]`

Purpose: To extract characters from the middle of a string.

x becomes the middle characters of y, starting from position w, up to z number of characters.

.find()

Example code: `x = y.find(z)`

Purpose: To find the position of substring z in y.

X becomes the position in the string y where z can be found. E.g. x = “Hello World”.find(”World”) returns 6 as the letter W is the sixth character in the string (starting at zero).

Qualification Accredited

OCR
Oxford Cambridge and RSA

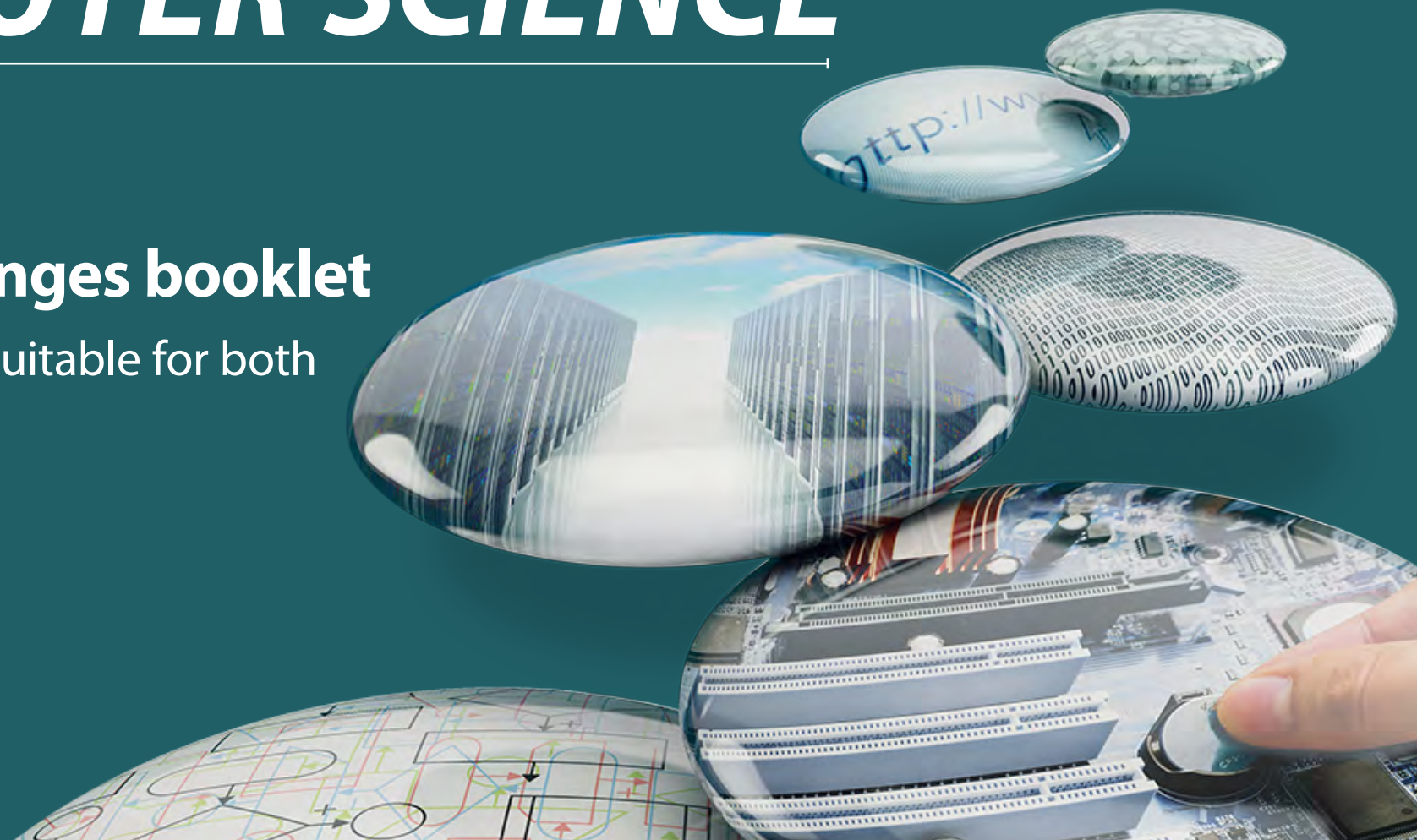
GCSE (9-1) and A LEVEL

COMPUTER SCIENCE

Coding challenges booklet

Coding challenges suitable for both GCSE and A Level

Version 1



www.ocr.org.uk/computerscience

Contents

CONTENTS

INTRODUCTION

These coding challenges provide real world problems for teachers and students to help develop their coding skills.

For GCSE: responses only need command line interfaces, and focus on using the programming techniques contained within the current specification.

For A Level: any solutions should have a graphical use interface created for it. Use of OOP methodologies is to be encouraged, as many problems lend themselves to a class system.

This is an active document and likely to receive regular updates with challenges throughout the lifetime of the specification.

We do not publish solutions, as there are many ways in which these problems could be solved. Discussions regarding approaches are beneficial at a cohort/class level to encourage candidate's realisation that each problem has many unique solutions that will fulfill the success criteria that have been identified. Where we do provide a solution - it should be used for discussion and comment, rather than being taken as 'the only and/or best way to solve the challenge'.

1	**Factorial Finder** The Factorial of a positive integer, n, is defined as the product of the sequence n, n-1, n-2, ...1 and the factorial of zero, 0, is defined as being 1. Solve this using both loops and recursion.
2	**Speed Tracker** Create a program that takes a time for a car going past a speed camera, the time going past the next one and the distance between them to calculate the average speed for the car in mph. The cameras are one mile apart. **Extensions:** 1. Speed cameras know the timings of each car going past, through number plate recognition. Valid number plates are two letters, two numbers and three letters afterwards, for example XX77 787. Produce a part of the program that checks whether a number plate matches the given pattern. Tell the user either way. 2. Create a program for creating a file of details for vehicles exceeding the speed limit set for a section of road. You will need to create a suitable file with test data, including randomised number plates and times. You will then use the code you've already written to process this list to determine who is breaking the speed limit (70mph) and who has invalid number plates.
3	**Thief!** A thief has managed to find out the four digits for an online PIN code, but doesn't know the correct sequence needed to hack into the account. Design and write a program that displays all the possible combinations for any four numerical digits entered by the user. The program should avoid displaying the same combination more than once. Submit a fully detailed Showcase for your program.
4	**Classification** A simple classification system asks a series of Yes/No questions in order to work out what type of animal is being looked at. Eg Does it have 4 legs? Does it eat meat? Does it have stripes? These systems can often be drawn using a "tree" structure. Carry out some simple research on classification trees, then write a program to help the user decide between the following: horse, cow, sheep, pig, dog, cat, lion, tiger, whale, dolphin, seal, penguin, ostrich, sparrow, spider, ant, bee, wasp, termite, octopus, squid Is there a better way to do this than using 101 IF...ELSE...END IFs? Develop your classification system for your own area of interest: pop bands; pokemon; cars; footballers; teachers; diseases etc.

5	**Fruit Machine** Write a program to simulate a Fruit Machine that displays three symbols at random from Cherry, Bell, Lemon, Orange, Star, Skull. The player starts with £1 credit, with each go costing 20p. If the Fruit Machine "rolls" two of the same symbol, the user wins 50p. The player wins £1 for three of the same and £5 for 3 Bells. The player loses £1 if two skulls are rolled and all of his/her money if three skulls are rolled. The player can choose to quit with the winnings after each roll or keep playing until there is no money left.
6	**Unit Converter (temp, currency, volume)** Converts various units between one another. The user enters the type of unit being entered, the type of unit they want to convert to and then the value. The program will then make the conversion.
7	**Credit Card Validator** Takes in a credit card number from a common credit card vendor (Visa, MasterCard, American Express, Discoverer) and validates it to make sure that it is a valid number (look into how credit cards use a checksum).
8	**Arithmetic test** A primary school teacher wants a computer program to test the basic arithmetic skills of her students. Generate random questions (2 numbers only) consisting of addition, subtraction, multiplication and division. The system should ask the student's name and then ask ten questions. The program should feed back if the answers are correct or not, and then generate a final score at the end. **Extensions:** 1. Extend your program so that it stores the results somewhere. The teacher has three classes, so you need to enable the program to distinguish between them. 2. The teacher wants to be able to log student performance in these tests. The teacher would like the program to store the last three scores for each student and to be able to output the results in alphabetical order with the student's highest score first out of the three.
9	**Happy Numbers** A happy number is defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers. Display an example of your output here. Find the first eight happy numbers.

10	**Number Names** Show how to spell out a number in English. You can use a pre-existing implementation or make your own, but you should support inputs up to at least one million (or the maximum value of your language's default bounded integer type, if that's less). **Extensions:** 1. Create support for inputs other than positive integers (like zero, negative integers, and floating-point numbers).
11	**Regex Query Tool** This is a tool that allows the user to enter a text string and then in a separate text box enter a regex pattern. It will run the regular expression against the string and return any matches or flag errors in the regular expression.
12	**Quiz Maker** Make an application which takes various questions from a file, picked randomly, and puts together a quiz for students. Each quiz can be different and then reads a key to grade the quizzes.
13	**Caesar Cipher** Implement a Caesar cipher, both encoding and decoding. The key is an integer from 1 to 25. This cipher rotates the letters of the alphabet (A to Z). The encoding replaces each letter with the 1st to 25th next letter in the alphabet (wrapping Z to A). So key 2 encrypts "HI" to "JK", but key 20 encrypts "HI" to "BC".
14	**Events calendar** Create a menu driven program that allows the user to add or delete events from a list of dates and timings, just like a calendar. The program should warn you if any of the events overlap when entering them. **Extensions:** 1. Make it so that none of the events are hard-coded into the program
15	**Pangrams** **"The quick brown fox jumps over the lazy dog";** note how all 26 English-language letters are used in the sentence. Your goal is to implement a program that takes a series of strings (one per line) and prints either True (the given string is a pangram), or False if it is not.
16	**Kaprekar** Determine whether a number is a Kaprekar number or not. See http://mathworld.wolfram.com/KaprekarNumber.html for more information.

17	**Number Table** Write a program that takes a symbol (+,-,* or /) and a natural number (>0) and makes a table like below for the operation from 0 to n For this example the user has entered "+ 4": + | 0 1 2 3 4 ---------- 0 | 0 1 2 3 4 1 | 1 2 3 4 5 2 | 2 3 4 5 6 3 | 3 4 5 6 7 4 | 4 5 6 7 8
18	**Years in a Range** Write a program to count the number years in a range that has a repeated digit. For example, 2012 has a repeated digit, but 2013 does not.
19	**Logic Gate** Write a program that will give the students the answer to logic gate questions For example: Enter logic gate : OR Enter first input : 1 Enter second input : 0 Result = 1 It should work for the logic gates OR, AND, XOR, NAND and NOR
20	**Palindromes** Write a program that checks if a string entered by the user is a palindrome. A palindrome is a word that reads the same forwards as backwards like "racecar"

21	**Data Entry** Create a program that retrieves the membership details for a Rock Climbing Club. The program should take a range of details and then repeat them back, with headings, for confirmation. Once confirmed, the program stores these details; else it clears them and allows a new input. **Extensions:** 1. Allow entry of more than one membership 2. Store membership details to a file 3. Retrieve details from a file 4. Allow searching for stored users
22	**Simple Life Calculator** Create a program that has 3 simple calculators within it, e.g. VAT, Tax and Times table. Allow users to choose which calculator they want to use and then carry out that calculation. **Extensions:** 1. Use a an option menu so that the user can use more than one calculation before the program closes
23	**Fibbing** Create a program that will calculate the Fibonacci Sequence to 10 places. **Extensions:** 1. Allow the user to specify the number of places generated 2. Print this in reverse order 3. Display the total of all the numbers shown

24	**Hack-proof** Create a program that will only open a text document if the correct password is entered. The user should choose the username and password first and it should also verify the password before allowing it. **Extensions:** 1. Create a random password first of at least 8 characters first as a suggested password 2. Create a random password that contains at least a lowercase, uppercase and special character of at least 8 characters in length 3. Verify that the password given by the user matches: a. The limits in Extension 1 above b. The limits in Extension 2 above
25	**Ordering** Create a program that allows entry of 10 numbers and then sorts them into ascending or descending order, based on user input. **Extension:** 1. The user can input a word or string, and it arranges the string into alphabetical order. E.g. My Rabbit would be shown as "abbimty". (Punctuation placement is not essential) 2. Repeat Extension 1, but include the sentence structure
26	**Truth or not** Create a program that would take the number of inputs in a logic circuit and works out the number of output lines are needed for the truth table. Have it draw the truth table on screen, using Columns for Inputs (A, B, C etc) and rows for the 1's and 0's. **Extension:** 1. Fill in the rest of the truth table if you can!
27	**Word Subtraction** Create a program that takes two strings/words. Then then converts this to an ASCII value and subtracts the values from each other. **Extension:** 1. Also add a function that removes any characters in the second word that occur in the first word. E.g. Fish and Tin, would return "Fsh" and "Tn"

28	**Name that Number** Telephone Keypads often have letters associated with each number. This means that 0141 117 2556 could be stored as 0141-CAT-DOOR. Create a program that can convert a phone number with "letters" into one that only contains digits. **Extensions:** 1. Can you develop your program so that only words in the dictionary are allowed?
29	**Item Merge** Create a program that will compare two shopping lists from "Week A" and "Week B". It will return any unique items contained on the list. **Extension:** 1. Append the two lists, with no repetition 2. Develop this to 4 Weeks of shopping an highlight the top 3 most popular items
30	**Year Addition** Create a program that accepts a year in the format ####, e.g. 2015. The program then adds each digit of the year together and outputs the answer. E.g. 2015 becomes the output 8. **Extension:** 1. Develop this so that the user can guess an integer value. If the MOD division is "0" they score a point, if it isn't they can guess again, up to 3 attempts in total
31	**Forwards and Backwards** Create a program that is able to detect if an input is the same as the reverse of the same input – i.e. a Palindrome
32	**Code it up** Create a program that adds 25 to the value of each character of a string that a user enters. This new string should be saved and output. **Extension:** 1. Develop your program to include a conversion from a 'coded' string back to a normal string 2. Develop your program to allow the user to enter the number they want the string coded by (e.g. 12) 3. Develop your program to then decode a string, based on the coded value that the end user enters

33	**Mor-se Coding** Create a program that allows you to enter a string and encode it into Morse code, using '.' and '-' notation. Spaces between words should be replaced with the "|" (pipe) character. Use a normal space for gaps between each character. **Extension:** 1. Develop your program to translate from Morse to alphanumeric, using the standards above
34	**What's the day?** Design a program to take 3 inputs, one for day, one for month and one for year. Get your program to validate if this is an actual day, and, if it is, output the day of the week it is! *Hint: How do leap years affect this program?*
35	**Game of Chance** A user can bet on any number from 0 to 30. If it's an even number they 2x their money back. If it's a multiple of 10 they get 3x their money back. If it's a prime number they get 5x their money back. If the number is below 5 they get a 2x bonus. Create a program that allows the user to guess a number. A random number is generated. If the guess == the random number then the user wins and gets a pay-out. Combinations of the win scenarios should be catered for.. e.g. 20 wins 2 x 3 bonus = 6x their money. **Extension:** 1. Develop your program to allow a user to enter the amount they want to place for that bet, and work out the resulting pay-out 2. Develop your program to store the user's current balance and stop them from betting if they have no money left 3. Develop your program to finally incorporate validation so that they cannot enter into a negative about of cash ever, and that a bet should be between 1 and 10 units of currency 4. Develop your program to allow multiple bets on different numbers
36	**Triangulate** Create a program that accepts 3 sides of a triangle. It then works out if these sides form a triangle, and if so, what type of triangle (e.g. Scalene, Isosceles, Right-Angle….) **Extension:** 1. Develop your program to allow 2 sides of a triangle and an angle, to work out the length of the missing side

37	**Fizz Buzz** Create a program that replicates the famous game Fizz Buzz. The program will take an input, e.g. 20, and then print out the list of Fizz Buzz up to and including that number, where: • Any multiple of 3 is replaced by the word 'Fizz' • Any multiple of 5 is replaced by the word 'Buzz' • Any multiple of both 3 and 5 is replaced by the word 'FizzBuzz' **Extension:** 1. Replace any prime number with the word 'OOPS!' 2. Allow the user to enter the base numbers that they want to replace words with. E.g. 2 and 3, which would mean: • Any multiple of 2 is replaced by the word 'Fizz' • Any multiple of 3 is replaced by the word 'Buzz' • Any multiple of both 2 and 3 is replaced by the word 'FizzBuzz'
38	**Sing Along** Create a program that prints the lyrics to the song '10 green bottles' in as few lines of code as possible. **Extension:** 1. Develop this program so that you can enter any starting number and it will count down from there
39	**Even more Odd** Create a program that accepts and random integer array (at least 10 integers) and orders them firstly by size, (small to large), and then puts all the even numbers AFTER the odd numbers within the array. It then echos the original array and the modified array to screen. E.g. an array 1,2,3,4,5,6,7,8,9,10 would be output 1,3,5,7,9,2,4,6,8,10. **Extension:** 1. Develop your program to allow Character input as well, and these come before the integers, and are listed in reverse alphabetical order
40	**Base of Numbers** Create a program that converts a denary number into its hexadecimal equivalent. **Extension:** 1. Allow the user to specify the base that they want to convert the number into, using an integer, e.g. 16 for Hexadecimal

The small print

We'd like to know your view on the resources we produce. By clicking on the 'Like' or 'Dislike' button you can help us to ensure that our resources work for you. When the email template pops up please add additional comments if you wish and then just click 'Send'. Thank you.

Whether you already offer OCR qualifications, are new to OCR, or are considering switching from your current provider/awarding organisation, you can request more information by completing the Expression of Interest form which can be found here: www.ocr.org.uk/expression-of-interest

Please get in touch if you want to discuss the accessibility of resources we offer to support delivery of our qualifications: resources.feedback@ocr.org.uk

Looking for a resource?

There is now a quick and easy search tool to help find **free** resources for your qualification:

www.ocr.org.uk/i-want-to/find-resources/

www.ocr.org.uk/alevelreform

OCR Customer Contact Centre

General qualifications

Telephone 01223 553998
Facsimile 01223 552627
Email general.qualifications@ocr.org.uk

OCR is part of Cambridge Assessment, a department of the University of Cambridge. *For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored.*

A DIVISION OF CAMBRIDGE ASSESSMENT

LLOYD'S REGISTER · LRQA ISO 9001 001

UKAS MANAGEMENT SYSTEMS

Qualification Accredited

OCR
Oxford Cambridge and RSA

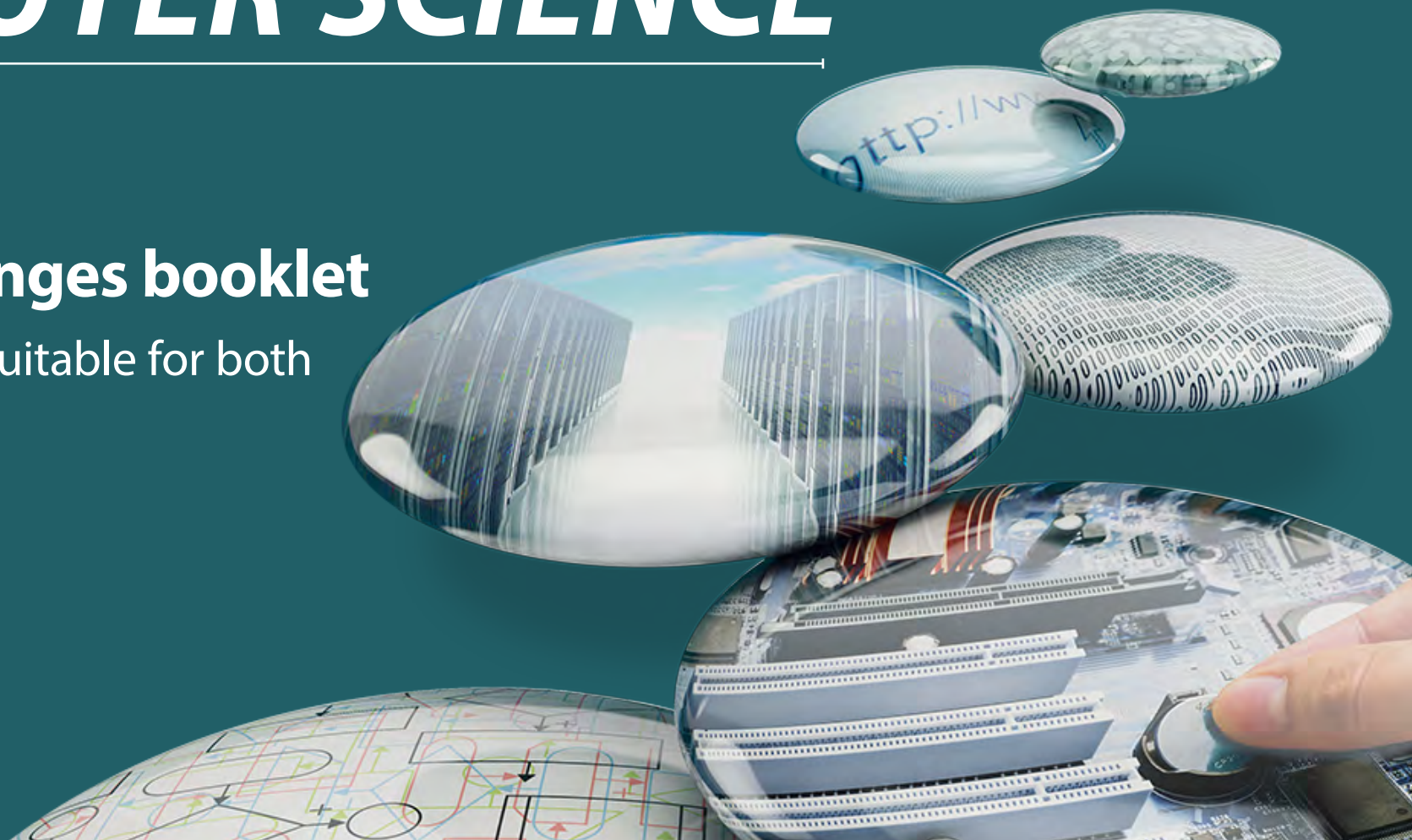
GCSE (9-1) and A LEVEL

COMPUTER SCIENCE

Coding challenges booklet

Coding challenges suitable for both GCSE and A Level

Version 2



www.ocr.org.uk/computerscience

Contents

CONTENTS

Contents

INTRODUCTION

These coding challenges provide real world problems for teachers and students to help develop their coding skills.

For GCSE: responses only need command line interfaces, and focus on using the programming techniques contained within the current specification.

For A Level: any solutions should have a graphical use interface created for it. Use of OOP methodologies is to be encouraged, as many problems lend themselves to a class system.

This is an active document and likely to receive regular updates with challenges throughout the lifetime of the specification.

We do not publish solutions, as there are many ways in which these problems could be solved. Discussions regarding approaches are beneficial at a cohort/class level to encourage candidate's realisation that each problem has many unique solutions that will fulfill the success criteria that have been identified. Where we do provide a solution - it should be used for discussion and comment, rather than being taken as 'the only and/or best way to solve the challenge'.

1	**Factorial Finder** The Factorial of a positive integer, n, is defined as the product of the sequence n, n-1, n-2, ...1 and the factorial of zero, 0, is defined as being 1. Solve this using both loops and recursion.
2	**Speed Tracker** Create a program that takes a time for a car going past a speed camera, the time going past the next one and the distance between them to calculate the average speed for the car in mph. The cameras are one mile apart. **Extensions:** 1. Speed cameras know the timings of each car going past, through number plate recognition. Valid number plates are two letters, two numbers and three letters afterwards, for example XX77 787. Produce a part of the program that checks whether a number plate matches the given pattern. Tell the user either way. 2. Create a program for creating a file of details for vehicles exceeding the speed limit set for a section of road. You will need to create a suitable file with test data, including randomised number plates and times. You will then use the code you've already written to process this list to determine who is breaking the speed limit (70mph) and who has invalid number plates.
3	**Thief!** A thief has managed to find out the four digits for an online PIN code, but doesn't know the correct sequence needed to hack into the account. Design and write a program that displays all the possible combinations for any four numerical digits entered by the user. The program should avoid displaying the same combination more than once. Submit a fully detailed Showcase for your program.
4	**Classification** A simple classification system asks a series of Yes/No questions in order to work out what type of animal is being looked at. Eg Does it have 4 legs? Does it eat meat? Does it have stripes? These systems can often be drawn using a "tree" structure. Carry out some simple research on classification trees, then write a program to help the user decide between the following: horse, cow, sheep, pig, dog, cat, lion, tiger, whale, dolphin, seal, penguin, ostrich, sparrow, spider, ant, bee, wasp, termite, octopus, squid Is there a better way to do this than using 101 IF...ELSE...END IFs? Develop your classification system for your own area of interest: pop bands; pokemon; cars; footballers; teachers; diseases etc.

5	**Fruit Machine** Write a program to simulate a Fruit Machine that displays three symbols at random from Cherry, Bell, Lemon, Orange, Star, Skull. The player starts with £1 credit, with each go costing 20p. If the Fruit Machine "rolls" two of the same symbol, the user wins 50p. The player wins £1 for three of the same and £5 for 3 Bells. The player loses £1 if two skulls are rolled and all of his/her money if three skulls are rolled. The player can choose to quit with the winnings after each roll or keep playing until there is no money left.
6	**Unit Converter (temp, currency, volume)** Converts various units between one another. The user enters the type of unit being entered, the type of unit they want to convert to and then the value. The program will then make the conversion.
7	**Credit Card Validator** Takes in a credit card number from a common credit card vendor (Visa, MasterCard, American Express, Discoverer) and validates it to make sure that it is a valid number (look into how credit cards use a checksum).
8	**Arithmetic test** A primary school teacher wants a computer program to test the basic arithmetic skills of her students. Generate random questions (2 numbers only) consisting of addition, subtraction, multiplication and division. The system should ask the student's name and then ask ten questions. The program should feed back if the answers are correct or not, and then generate a final score at the end. **Extensions:** 1. Extend your program so that it stores the results somewhere. The teacher has three classes, so you need to enable the program to distinguish between them. 2. The teacher wants to be able to log student performance in these tests. The teacher would like the program to store the last three scores for each student and to be able to output the results in alphabetical order with the student's highest score first out of the three.
9	**Happy Numbers** A happy number is defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers. Display an example of your output here. Find the first eight happy numbers.

10	**Number Names** Show how to spell out a number in English. You can use a pre-existing implementation or make your own, but you should support inputs up to at least one million (or the maximum value of your language's default bounded integer type, if that's less). **Extensions:** 1. Create support for inputs other than positive integers (like zero, negative integers, and floating-point numbers).
11	**Regex Query Tool** This is a tool that allows the user to enter a text string and then in a separate text box enter a regex pattern. It will run the regular expression against the string and return any matches or flag errors in the regular expression.
12	**Quiz Maker** Make an application which takes various questions from a file, picked randomly, and puts together a quiz for students. Each quiz can be different and then reads a key to grade the quizzes.
13	**Caesar Cipher** Implement a Caesar cipher, both encoding and decoding. The key is an integer from 1 to 25. This cipher rotates the letters of the alphabet (A to Z). The encoding replaces each letter with the 1st to 25th next letter in the alphabet (wrapping Z to A). So key 2 encrypts "HI" to "JK", but key 20 encrypts "HI" to "BC".
14	**Events calendar** Create a menu driven program that allows the user to add or delete events from a list of dates and timings, just like a calendar. The program should warn you if any of the events overlap when entering them. **Extensions:** 1. Make it so that none of the events are hard-coded into the program
15	**Pangrams** **"The quick brown fox jumps over the lazy dog";** note how all 26 English-language letters are used in the sentence. Your goal is to implement a program that takes a series of strings (one per line) and prints either True (the given string is a pangram), or False if it is not.
16	**Kaprekar** Determine whether a number is a Kaprekar number or not. See http://mathworld.wolfram.com/KaprekarNumber.html for more information.

17	**Number Table** Write a program that takes a symbol (+,-,* or /) and a natural number (>0) and makes a table like below for the operation from 0 to n For this example the user has entered "+ 4": + | 0 1 2 3 4 ---------- 0 | 0 1 2 3 4 1 | 1 2 3 4 5 2 | 2 3 4 5 6 3 | 3 4 5 6 7 4 | 4 5 6 7 8
18	**Years in a Range** Write a program to count the number years in a range that has a repeated digit. For example, 2012 has a repeated digit, but 2013 does not.
19	**Logic Gate** Write a program that will give the students the answer to logic gate questions For example: Enter logic gate : OR Enter first input : 1 Enter second input : 0 Result = 1 It should work for the logic gates OR, AND, XOR, NAND and NOR
20	**Palindromes** Write a program that checks if a string entered by the user is a palindrome. A palindrome is a word that reads the same forwards as backwards like "racecar"

21	**Data Entry** Create a program that retrieves the membership details for a Rock Climbing Club. The program should take a range of details and then repeat them back, with headings, for confirmation. Once confirmed, the program stores these details; else it clears them and allows a new input. **Extensions:** 1. Allow entry of more than one membership 2. Store membership details to a file 3. Retrieve details from a file 4. Allow searching for stored users
22	**Simple Life Calculator** Create a program that has 3 simple calculators within it, e.g. VAT, Tax and Times table. Allow users to choose which calculator they want to use and then carry out that calculation. **Extensions:** 1. Use a an option menu so that the user can use more than one calculation before the program closes
23	**Fibbing** Create a program that will calculate the Fibonacci Sequence to 10 places. **Extensions:** 1. Allow the user to specify the number of places generated 2. Print this in reverse order 3. Display the total of all the numbers shown

24	**Hack-proof** Create a program that will only open a text document if the correct password is entered. The user should choose the username and password first and it should also verify the password before allowing it. **Extensions:** 1. Create a random password first of at least 8 characters first as a suggested password 2. Create a random password that contains at least a lowercase, uppercase and special character of at least 8 characters in length 3. Verify that the password given by the user matches: a. The limits in Extension 1 above b. The limits in Extension 2 above
25	**Ordering** Create a program that allows entry of 10 numbers and then sorts them into ascending or descending order, based on user input. **Extension:** 1. The user can input a word or string, and it arranges the string into alphabetical order. E.g. My Rabbit would be shown as "abbimty". (Punctuation placement is not essential) 2. Repeat Extension 1, but include the sentence structure
26	**Truth or not** Create a program that would take the number of inputs in a logic circuit and works out the number of output lines are needed for the truth table. Have it draw the truth table on screen, using Columns for Inputs (A, B, C etc) and rows for the 1's and 0's. **Extension:** 1. Fill in the rest of the truth table if you can!
27	**Word Subtraction** Create a program that takes two strings/words. Then then converts this to an ASCII value and subtracts the values from each other. **Extension:** 1. Also add a function that removes any characters in the second word that occur in the first word. E.g. Fish and Tin, would return "Fsh" and "Tn"

28	**Name that Number** Telephone Keypads often have letters associated with each number. This means that 0141 117 2556 could be stored as 0141-CAT-DOOR. Create a program that can convert a phone number with "letters" into one that only contains digits. **Extensions:** 1. Can you develop your program so that only words in the dictionary are allowed?
29	**Item Merge** Create a program that will compare two shopping lists from "Week A" and "Week B". It will return any unique items contained on the list. **Extension:** 1. Append the two lists, with no repetition 2. Develop this to 4 Weeks of shopping an highlight the top 3 most popular items
30	**Year Addition** Create a program that accepts a year in the format ####, e.g. 2015. The program then adds each digit of the year together and outputs the answer. E.g. 2015 becomes the output 8. **Extension:** 1. Develop this so that the user can guess an integer value. If the MOD division is "0" they score a point, if it isn't they can guess again, up to 3 attempts in total
31	**Forwards and Backwards** Create a program that is able to detect if an input is the same as the reverse of the same input – i.e. a Palindrome
32	**Code it up** Create a program that adds 25 to the value of each character of a string that a user enters. This new string should be saved and output. **Extension:** 1. Develop your program to include a conversion from a 'coded' string back to a normal string 2. Develop your program to allow the user to enter the number they want the string coded by (e.g. 12) 3. Develop your program to then decode a string, based on the coded value that the end user enters

33	**Mor-se Coding** Create a program that allows you to enter a string and encode it into Morse code, using '.' and '-' notation. Spaces between words should be replaced with the "|" (pipe) character. Use a normal space for gaps between each character. **Extension:** 1. Develop your program to translate from Morse to alphanumeric, using the standards above
34	**What's the day?** Design a program to take 3 inputs, one for day, one for month and one for year. Get your program to validate if this is an actual day, and, if it is, output the day of the week it is! *Hint: How do leap years affect this program?*
35	**Game of Chance** A user can bet on any number from 0 to 30. If it's an even number they 2x their money back. If it's a multiple of 10 they get 3x their money back. If it's a prime number they get 5x their money back. If the number is below 5 they get a 2x bonus. Create a program that allows the user to guess a number. A random number is generated. If the guess == the random number then the user wins and gets a pay-out. Combinations of the win scenarios should be catered for.. e.g. 20 wins 2 x 3 bonus = 6x their money. **Extension:** 1. Develop your program to allow a user to enter the amount they want to place for that bet, and work out the resulting pay-out 2. Develop your program to store the user's current balance and stop them from betting if they have no money left 3. Develop your program to finally incorporate validation so that they cannot enter into a negative about of cash ever, and that a bet should be between 1 and 10 units of currency 4. Develop your program to allow multiple bets on different numbers
36	**Triangulate** Create a program that accepts 3 sides of a triangle. It then works out if these sides form a triangle, and if so, what type of triangle (e.g. Scalene, Isosceles, Right-Angle….) **Extension:** 1. Develop your program to allow 2 sides of a triangle and an angle, to work out the length of the missing side

37	**Fizz Buzz** Create a program that replicates the famous game Fizz Buzz. The program will take an input, e.g. 20, and then print out the list of Fizz Buzz up to and including that number, where: • Any multiple of 3 is replaced by the word 'Fizz' • Any multiple of 5 is replaced by the word 'Buzz' • Any multiple of both 3 and 5 is replaced by the word 'FizzBuzz' **Extension:** 1. Replace any prime number with the word 'OOPS!' 2. Allow the user to enter the base numbers that they want to replace words with. E.g. 2 and 3, which would mean: • Any multiple of 2 is replaced by the word 'Fizz' • Any multiple of 3 is replaced by the word 'Buzz' • Any multiple of both 2 and 3 is replaced by the word 'FizzBuzz'
38	**Sing Along** Create a program that prints the lyrics to the song '10 green bottles' in as few lines of code as possible. **Extension:** 1. Develop this program so that you can enter any starting number and it will count down from there
39	**Even more Odd** Create a program that accepts and random integer array (at least 10 integers) and orders them firstly by size, (small to large), and then puts all the even numbers AFTER the odd numbers within the array. It then echos the original array and the modified array to screen. E.g. an array 1,2,3,4,5,6,7,8,9,10 would be output 1,3,5,7,9,2,4,6,8,10. **Extension:** 1. Develop your program to allow Character input as well, and these come before the integers, and are listed in reverse alphabetical order
40	**Base of Numbers** Create a program that converts a denary number into its hexadecimal equivalent. **Extension:** 1. Allow the user to specify the base that they want to convert the number into, using an integer, e.g. 16 for Hexadecimal

41	**Prime Factorisation** Have the user enter a number and find all Prime Factors (if there are any) and display them. **Extension:** • Have the program find prime numbers until the user chooses to stop asking for the next one.
42	**Tilers mate** Have the user enter the Width and Length of the floor and have the program calculate the total cost of tiles it would take to cover a floor plan using a cost entered by the user (per tile or metre^2). **Extension:** • Have the programme offer different types of tiles with different costs and tell the user the cost. • Have the programme take into account the cost of grout and labour to give a customer a quote with and without VAT.
43	**The meaning of life** Have the program make an animation of the game of life (https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life). **Extension:** • Let the user set up the initial state Let the user change cells while the animation is running
44	**Sudoku** Have the program solve a Sudoku (https://en.wikipedia.org/wiki/Sudoku).
45	**Find the factorial** The Factorial of a positive integer, n, is defined as the product of the sequence n, n-1, n-2, ...1 and the factorial of zero, 0, is defined as being 1. Solve this using both loops and recursion.
46	**Complex Numbers** Have the programme show addition, multiplication, negation, and inversion of complex numbers in separate functions. (Subtraction and division operations can be made with pairs of these operations.) Print the results for each operation tested to screen.
47	**Happy Numbers =)** A happy number is defined by the following process. Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers. Have the programme find the first 8 happy numbers.

48	**Reverse it** Have the programme allow a user to enter some text and then the programme will reverse it and print it back to the screen. **Extension:** • Have the programme count the vowels and consonants and print these to screen. • Have the programme check if the text is a palindrome (it is the same forwards as it is backwards e.g "racecar" or "hannah").
49	**Fireworks** Make an animation of a firework display, with rockets, Catherine wheels etc. **Extension:** • Let the user specify the number, colour, timing and location of fireworks.
50	**Mandelbrot Set** Draw a Mandelbrot set (http://mathworld.wolfram.com/MandelbrotSet.html). **Extension:** • In colour • With an animation • Allow the user to zoom in
51	**Text-speak converter** Set up a text-speak to English dictionary and have the program convert input from text-speak to English. ("lol" to "laugh out loud" etc) **Extension:** • Read in the text from a file Allow the user to add new entries in the dictionary
52	**Is this card valid?** Have the programme take in a credit card number from a common credit card vendor (Visa, MasterCard, American Express, Discoverer) and validates it to make sure that it is a valid number (look into how credit cards use a checksum). ***Don't use any real card details***

53	**Mortgage Calculator** Have the programme calculate the monthly payments of a fixed term mortgage over given Nth terms at a given interest rate. Also figure out how long it will take the user to pay back the loan. **Extension:** • Add an option for users to select the compounding interval (Monthly, Weekly, Daily, Continually). • Add in functionality to deal with over payments at a given % each month.
54	**Dear Diary** Have the programme allow people to add comments or write diary entries. It should add timestamps to all entries. Could also be made into a shout box (https://en.wikipedia.org/wiki/Shoutbox). **Extension:** • Add date stamps to each post. • Embed your programme in a webpage. • Have the programme save the diary entries externally.
55	**Secret Ciphers** Have the programme encrypt messages using one of the following ciphers: • Vigenere • Vernan • Ceasar The cipher can ignore numbers, symbols and whitespace. **Extension:** • Create separate functions for each Cipher and allow the user to choose which one to use. • Have the programme decrypt messages also. • Have the programme email the encrypted message to a friend.

56	**Page Scraper** Have the programme connect to a site and pulls out all the links, or images, and save them to a list. **Extension:** • Organize the indexed content and don't allow duplicates. • Have it put the results into an easily searchable index file.
57	**Such meme, many like** Have the programme display a blank meme template (https://imgflip.com/memetemplates) and allow the user to add their own text (don't forget to use Impact as the font). **Extension:** • Display the finished memes in a webpage • Have suggestions about the different memes on how to write them properly
58	**Text based game** Create a text based game like Zork. Have the programme take the users name and use it to tell the story. Have the programme allow a non-linear progression through the various rooms. The user should have at least 2 choices per room. You will want to plan this one with a flowchart first! **Extension:** • Have each room as a separate function. • Add an inventory system so you must collect a key to get through a certain door etc.
59	**CSV File Utility** Have the programme read a .CSV file of records, sort them, and then write them back to the file. Allow the user to choose various sorting algorithms based on a chosen field.

60	**Get GIFy with it** Have the programme convert small video files into GIFs. Have a look at: http://zulko.github.io/blog/2014/01/23/making-animated-gifs-from-video-files-with-python/ for a tutorial. **Extension:** • Crop the video • Freeze a region • Make your GIF time symmetrical • Add some text • Remove the background
61	**Your name is…** Have the programme ask for your name, age and form. Have it tell them the information back in the format: Your name is (blank), you are (blank) years old, and you are in form (blank). **Extension:** • Have the programme store this information in an external file
62	**R@nd0m P@ssw0rd generator** Have the programme create random strong passwords mixing upper and lower case, symbols and numbers. **Extension:** • Have the password also use ASCII characters • Have the passwords stored in an external file
63	**I like Pi** Have the programme calculate pi to at least 30 decimal places.
64	**Galaxy song** Use graphics and random functions to draw an imaginary night sky filled with stars. **Extension:** • Add the milky way

65	**Spam filter** Take a list of dishes from a menu and add "spam" to them. See https://en.wikipedia.org/wiki/Spam_(Monty_Python). **Extension:** • Experiment with adding spam at the beginning, end and all places in-between
66	**Silly walks** Draw a random walk where each step of equal length is either up, down, right or left with equal probability. See https://en.wikipedia.org/wiki/Random_walk. **Extension:** • Model Brownian motion (https://en.wikipedia.org/wiki/Brownian_motion) in 2 dimensions with several particles • Fill your screen with a space filling curve (http://mathworld.wolfram.com/Plane-FillingFunction.html)
67	**What have the Romans ever done for us?** Have the user enter a number and print it out in Roman numerals.
68	**Semaphore** Have the user enter some text and make an animation of it converted into semaphore (https://en.wikipedia.org/wiki/Flag_semaphore).
69	**Beautiful soup** Use the BeautifulSoup and requests Python packages to print out a list of all the article titles on the BBC News (http://www.bbc.co.uk/news) This may help: http://www.pythonforbeginners.com/beautifulsoup/beautifulsoup-4-python
70	**Of mice and men** Have the programme allow a user to play the "mice and men" game. The game works like this: Randomly generate a 4-digit number. Ask the user to guess a 4-digit number. For every digit that the user guessed correctly in the correct place, they have a "mouse". For every digit the user guessed correctly in the wrong place is a "man" Every time the user makes a guess, tell them how many "mice" and "men" they have. Once the user guesses the correct number, the game is over. Keep track of the number of guesses the user makes throughout the game and tell the user at the end. **Extension:** • Deal with "mice" and "mouse" and "man" and "men " properly.

71	**Goldbach** Goldbach's conjecture says that every positive even number greater than 2 is the sum of two prime numbers. Example: 28 = 5 + 23. It is one of the most famous facts in number theory that has not been proved to be correct in the general case. It has been numerically confirmed up to very large. Write a predicate to find the two prime numbers that sum up to a given even integer.
72	**Lists** Create a list containing all integers within a given range. Insert an element at a given position into a list. Extract a given number of randomly selected elements from a list and create a list of lists. Sort the list of lists according to the length of sublists.
73	**Travel club** A group of people are member of a travel club. The group shares expenses equally but it is not practical to share every expense as they happen, so all expenses are collated (such as taxis, train tickets etc) after the trip and the member cost is shared to within 1% between the group. Create a programme that computes the net cost from a list of expenses and works out the minimum amount of money that must change hands in order for everybody to have paid the same amount (within 1%).

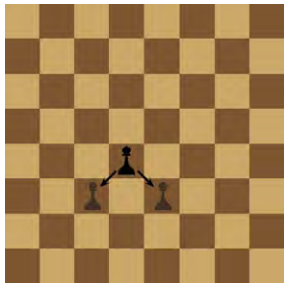
74

Checkmate checker

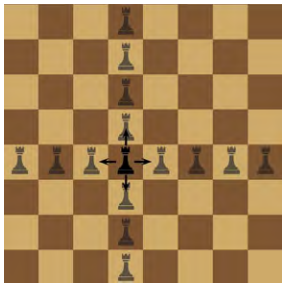
Create a programme that checks whether a King is in check in a given chess game configuration.

Movement examples are shown below, grey pieces indicate the positions where the piece can capture another piece:

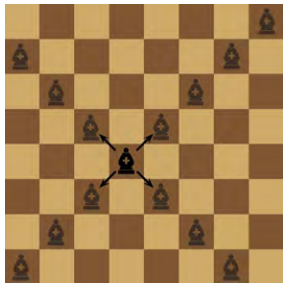
Pawn



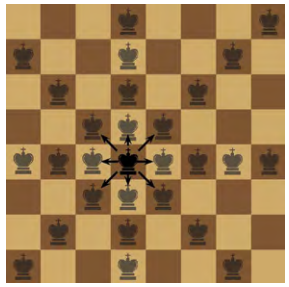
Castle



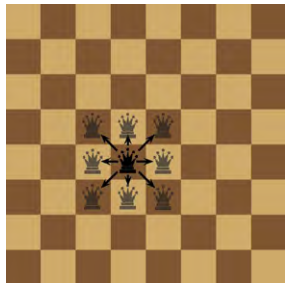
Bishop



Queen



King



Knight

There will be an arbitrary number of board configurations in the input, each consisting of eight lines of eight characters each. A ``.'' denotes an empty square, while upper- and lowercase letters represent the pieces as defined above. There will be no invalid characters and no configurations where both kings are in check. You must read until you find an empty board consisting only of ``.'' characters, which should not be processed. There will be an empty line between each pair of board configurations. All boards, except for the empty one, will contain exactly one white king and one black king.

For each board configuration read you must output one of the following answers:

Game #d: white king is in check.

Game #d: black king is in check.

Game #d: no king is in check.

where d stands for the game number starting from 1.

75

String permutation

Given two strings **x** and **y**, print the longest string a of letters such that there is a permutation of a that is a subsequence of **x** and there is a permutation of a that is a subsequence of **y**.

76	**That's a lot of number**

Work out the first ten digits of the sum of the following one-hundred 50-digit numbers.

37107287533902102798797998220837590246510135740250
46376937677490009712648124896970078050417018260538
74324986199524741059474233309513058123726617309629
91942213363574161572522430563301811072406154908250
23067588207539346171171980310421047513778063246676
89261670696623633820136378418383684178734361726757
28112879812849979408065481931592621691275889832738
44274228917432520321923589422876796487670272189318
47451445736001306439091167216856844588711603153276
70386486105843025439939619828917593665686757934951
62176457141856560629502157223196586755079324193331
64906352462741904929101432445813822663347944758178
92575867718337217661963751590579239728245598838407
58203565325359399008402633568948830189458628227828
80181199384826282014278194139940567587151170094390
35398664372827112653829987240784473053190104293586
86515506006295864861532075273371959191420517255829
71693888707715466499115593487603532921714970056938
54370070576826684624621495650076471787294438377604
53282654108756828443191190634694037855217779295145
36123272525000296071075082563815656710885258350721
45876576172410976447339110607218265236877223636045
17423706905851860660448207621209813287860733969412

	81142660418086830619328460811191061556940512689692
	51934325451728388641918047049293215058642563049483
	62467221648435076201727918039944693004732956340691
	15732444386908125794514089057706229429197107928209
	55037687525678773091862540744969844508330393682126
	18336384825330154686196124348767681297534375946515
	80386287592878490201521685554828717201219257766954
	78182833757993103614740356856449095527097864797581
	16726320100436897842553539920931837441497806860984
	48403098129077791799088218795327364475675590848030
	87086987551392711854517078544161852424320693150332
	59959406895756536782107074926966537676326235447210
	69793950679652694742597709739166693763042633987085
	41052684708299085211399427365734116182760315001271
	65378607361501080857009149939512557028198746004375
	35829035317434717326932123578154982629742552737307
	94953759765105305946966067683156574377167401875275
	88902802571733229619176668713819931811048770190271
	25267680276078003013678680992525463401061632866526
	36270218540497705585629946580636237993140746255962
	24074486908231174977792365466257246923322810917141
	91430288197103288597806669760892938638285025333403
	34413065578016127815921815005561868836468420090470
	23053081172816430487623791969842487255036638784583
	11487696932154902810424020138335124462181441773470
	63783299490636259666498587618221225225512486764533
	67720186971698544312419572409913959008952310058822
	95548255300263520781532296796249481641953868218774

760853271322857231104248034561248676970645079952 36

377742425354112916842768655389262050249103265729 67

237019132757256752856532482582654630922070585965 22

297988602722583319131263751473419948895347657455 01

184957014548792889848568277260777137214037988797 15

382982037830314735277215803481445134914737226651 381

348295438291999181802789165224310273922511228695 39

409579530664052326325380441000596549391598795936 35

297461521855023713076422551211836938035803885849 03

416981162220729771861582366784246891579935329619 22

624679571944012690438771072750481023908955235974 57

231897067725479150615055049539229795309011299675 19

861880882258753145295840992512038290094077707756 72

113067397083047244838165338735023408456470580773 08

829591747671403631980081871290118754913105471265 81

976233310448183862695154563349263665728975634005 00

428462801835170705278318394258821455212272512503 27

551216035469812005817621652128276527516912968977 89

322381957343293399464375019078369457658833523998 86

755061649651847751807381688378610915273579297013 37

621778427521926234019423996391680449839931733127 31

329241857071473495669166746876346609150359146775 04

995186714302352196288948901024233251169136196266 22

732674608005915474718307983928685352069469445407 24

768418225246744171615140364279822733480555562148 18

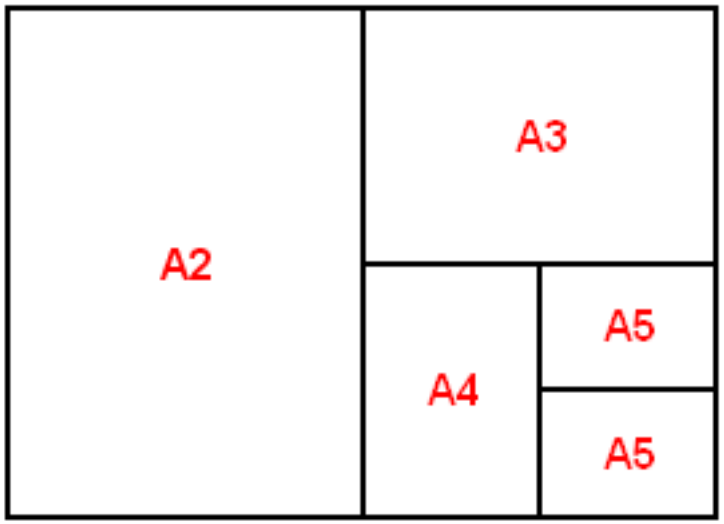
971426179103425986472045168939894221798260880768 52

877836461827993463137677543078093633330189826420 90

108488025216746708832151201858835432238128769527 86

	713296124747824645386369930090493103636197638780 39
	621840735723997942234062353938083396513274080111 16
	666278919814880877979418768761442300309844908514 11
	606618262936828367647447792391803351109890697907 14
	857869440895529906536404474255760836599766457950 96
	660243964099053896071201982199760475994901972302 97
	649139826800329731560371200413779037855660850892 52
	167309393198727502754689069037075394130426523150 11
	948093772450487951509541009216458637547105984367 91
	786391670211874924319957006419179697775990283006 99
	153687137119366149528113058763802784107544497330 78
	407899231155355625611423224232550336854424889173 53
	448899115014406480203690680639606723221932041495 35
	415031288803395360532993403680069777106505666319 54
	812348806732101467390585685579345814036278227032 80
	826165707739483275922328459417065250945123252306 08
	229188020587773197198394501808880724296619808111 97
	771585425020165450904132458097868827789487218596 17
	721078384350691861554356628840622574736922845095 16
	208496039801340017239306716668235552452528046097 22
	535035342264725242508740540755917897812643303316 90

77	**Fib on a chi** The Fibonacci sequence is defined by the recurrence relation: Fn = Fn–1 + Fn–2, where F1 = 1 and F2 = 1. Hence the first 12 terms will be: F1 = 1 F2 = 1 F3 = 2 F4 = 3 F5 = 5 F6 = 8 F7 = 13 F8 = 21 F9 = 34 F10 = 55 F11 = 89 F12 = 144 The 12th term, F12, is the first term to contain three digits. What is the index of the first term in the Fibonacci sequence to contain 1000 digits?
78	**2 fiddy** It is possible to make £2.50 in the following way: 1×£1 + 2×50p + 2×20p + 1×5p + 1×2p + 3×1p Write a programme that works out all the different ways £2.50 can be made using any number of coins.

79	**Printer problems** A printing shop runs 16 batches (jobs) every week and each batch requires a sheet of special colour-proofing paper of size A5. Every Monday morning, the foreman opens a new envelope, containing a large sheet of the special paper with size A1. He proceeds to cut it in half, thus getting two sheets of size A2. Then he cuts one of them in half to get two sheets of size A3 and so on until he obtains the A5-size sheet needed for the first batch of the week. All the unused sheets are placed back in the envelope. At the beginning of each subsequent batch, he takes from the envelope one sheet of paper at random. If it is of size A5, he uses it. If it is larger, he repeats the 'cut-in-half' procedure until he has what he needs and any remaining sheets are always placed back in the envelope. A1: 
80	**Happy Hopper** A sequence of n > 0 integers is called a happy hopper if the absolute values of the differences between successive elements take on all possible values 1 through n - 1. E.g 1 4 2 3 is a happy hopper because the absolute differences are 3, 2, and 1, respectively. The definition implies that any sequence of a single integer is a happy hopper. Write a program to determine whether each of a number of sequences is a happy hopper.

COMPUTER SCIENCE ALPHABET

All the links below lead to rosettacode.org and have a challenge and then a solution for the vast majority of programming languages. Content is available under GNU Free Documentation License 1.2 unless otherwise noted.

A	Abstract type	http://rosettacode.org/wiki/Abstract_type
	Apply a callback to an array	http://rosettacode.org/wiki/Apply_a_callback_to_an_array
	Arithmetic/Integer	http://rosettacode.org/wiki/Arithmetic/Integer
	Arrays	http://rosettacode.org/wiki/Arrays
	Assertions	http://rosettacode.org/wiki/Assertions
	Associative array/Creation	http://rosettacode.org/wiki/Associative_array/Creation
	Associative array/Iteration	http://rosettacode.org/wiki/Associative_array/Iteration
B	Binary digits	http://rosettacode.org/wiki/Binary_digits
	Boolean values	http://rosettacode.org/wiki/Boolean_values
C	Call an object method	http://rosettacode.org/wiki/Call_an_object_method
	Character codes	http://rosettacode.org/wiki/Character_codes
	Classes	http://rosettacode.org/wiki/Classes
	Collections	http://rosettacode.org/wiki/Collections
	Command-line arguments	http://rosettacode.org/wiki/Command-line_arguments
	Comments	http://rosettacode.org/wiki/Comments
	Compound data type	http://rosettacode.org/wiki/Compound_data_type
	Concurrent computing	http://rosettacode.org/wiki/Concurrent_computing
	Copy a string	http://rosettacode.org/wiki/Copy_a_string
	Count in octal	http://rosettacode.org/wiki/Count_in_octal
	Create a two-dimensional array at runtime	http://rosettacode.org/wiki/Create_a_two-dimensional_array_at_runtime
	Constants	http://rosettacode.org/wiki/Enforced_immutability
D	Define a primitive data type	http://rosettacode.org/wiki/Define_a_primitive_data_type
E	Empty program	http://rosettacode.org/wiki/Empty_program
	Enumerations	http://rosettacode.org/wiki/Enumerations

F	Factors of an integer	http://rosettacode.org/wiki/Factors_of_an_integer
	Filter	http://rosettacode.org/wiki/Filter
	Find limit of recursion	http://rosettacode.org/wiki/Find_limit_of_recursion
	Formatted numeric output	http://rosettacode.org/wiki/Formatted_numeric_output
	Function definition	http://rosettacode.org/wiki/Function_definition
G	Generic swap	http://rosettacode.org/wiki/Generic_swap
H	Hash from two arrays	http://rosettacode.org/wiki/Hash_from_two_arrays
	Hello world/Graphical	http://rosettacode.org/wiki/Hello_world/Graphical
	Hello world/Newline omission	http://rosettacode.org/wiki/Hello_world/Newline_omission
	Hello world/Standard error	http://rosettacode.org/wiki/Hello_world/Standard_error
	Hello world/Text	http://rosettacode.org/wiki/Hello_world/Text
I	Implicit type conversion	http://rosettacode.org/wiki/Implicit_type_conversion
	Include a file	http://rosettacode.org/wiki/Include_a_file
	Infinity	http://rosettacode.org/wiki/Infinity
	Inheritance/Multiple	http://rosettacode.org/wiki/Inheritance/Multiple
	Inheritance/Single	http://rosettacode.org/wiki/Inheritance/Single
	Input/user	http://rosettacode.org/wiki/User_input/Text
	Input loop	http://rosettacode.org/wiki/Input_loop
	Interactive programming	http://rosettacode.org/wiki/Interactive_programming
L	List comprehensions	http://rosettacode.org/wiki/List_comprehensions
	Literals/Floating point	http://rosettacode.org/wiki/Literals/Floating_point
	Literals/Integer	http://rosettacode.org/wiki/Literals/Integer
	Literals/String	http://rosettacode.org/wiki/Literals/String
M	Memory allocation	http://rosettacode.org/wiki/Memory_allocation
	Modulinos	http://rosettacode.org/wiki/Modulinos
	Multiline shebang	http://rosettacode.org/wiki/Multiline_shebang
N	Named parameters	http://rosettacode.org/wiki/Named_parameters
	Native shebang	http://rosettacode.org/wiki/Native_shebang
	Null object	http://rosettacode.org/wiki/Null_object
O	Optional parameters	http://rosettacode.org/wiki/Optional_parameters

P	Parametric polymorphism	http://rosettacode.org/wiki/Parametric_polymorphism
	Parse command-line arguments	http://rosettacode.org/wiki/Parse_command-line_arguments
	Pick random element	http://rosettacode.org/wiki/Pick_random_element
	Polymorphism	http://rosettacode.org/wiki/Polymorphism
	Program name	http://rosettacode.org/wiki/Program_name
	Program termination	http://rosettacode.org/wiki/Program_termination
R	Random numbers	http://rosettacode.org/wiki/Random_numbers
	Real constants and functions	http://rosettacode.org/wiki/Real_constants_and_functions
	Return multiple values	http://rosettacode.org/wiki/Return_multiple_values
S	Scope modifiers	http://rosettacode.org/wiki/Scope_modifiers
	Scope/Function names and labels	http://rosettacode.org/wiki/Scope/Function_names_and_labels
	Simple windowed application	http://rosettacode.org/wiki/Simple_windowed_application
	Sleep	http://rosettacode.org/wiki/Sleep
	Special characters	http://rosettacode.org/wiki/Special_characters
	Special variables	http://rosettacode.org/wiki/Special_variables
	Start from a main routine	http://rosettacode.org/wiki/Start_from_a_main_routine
	String append	http://rosettacode.org/wiki/String_append
	String comparison	http://rosettacode.org/wiki/String_comparison
	String concatenation	http://rosettacode.org/wiki/String_concatenation
	String interpolation (included)	http://rosettacode.org/wiki/String_interpolation_(included)
	String length	http://rosettacode.org/wiki/String_length
	String prepend	http://rosettacode.org/wiki/String_prepend
	Strip comments from a string	http://rosettacode.org/wiki/Strip_comments_from_a_string
	Substring	http://rosettacode.org/wiki/Substring
T	Take notes on the command line	http://rosettacode.org/wiki/Take_notes_on_the_command_line
	Topic variable	http://rosettacode.org/wiki/Topic_variable
U	User input/Graphical	http://rosettacode.org/wiki/User_input/Graphical
	User input/Text	http://rosettacode.org/wiki/User_input/Text
V	Variables	http://rosettacode.org/wiki/Variables
	Variadic function	http://rosettacode.org/wiki/Variadic_function

The small print

We'd like to know your view on the resources we produce. By clicking on the 'Like' or 'Dislike' button you can help us to ensure that our resources work for you. When the email template pops up please add additional comments if you wish and then just click 'Send'. Thank you.

Whether you already offer OCR qualifications, are new to OCR, or are considering switching from your current provider/awarding organisation, you can request more information by completing the Expression of Interest form which can be found here: www.ocr.org.uk/expression-of-interest

Please get in touch if you want to discuss the accessibility of resources we offer to support delivery of our qualifications: resources.feedback@ocr.org.uk

Looking for a resource?

There is now a quick and easy search tool to help find **free** resources for your qualification:

www.ocr.org.uk/i-want-to/find-resources/

www.ocr.org.uk/alevelreform

OCR Customer Contact Centre

General qualifications

Telephone 01223 553998
Facsimile 01223 552627
Email general.qualifications@ocr.org.uk

OCR is part of Cambridge Assessment, a department of the University of Cambridge. *For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored.*

A DIVISION OF CAMBRIDGE ASSESSMENT

LLOYD'S REGISTER · LRQA ISO 9001 001

UKAS MANAGEMENT SYSTEMS