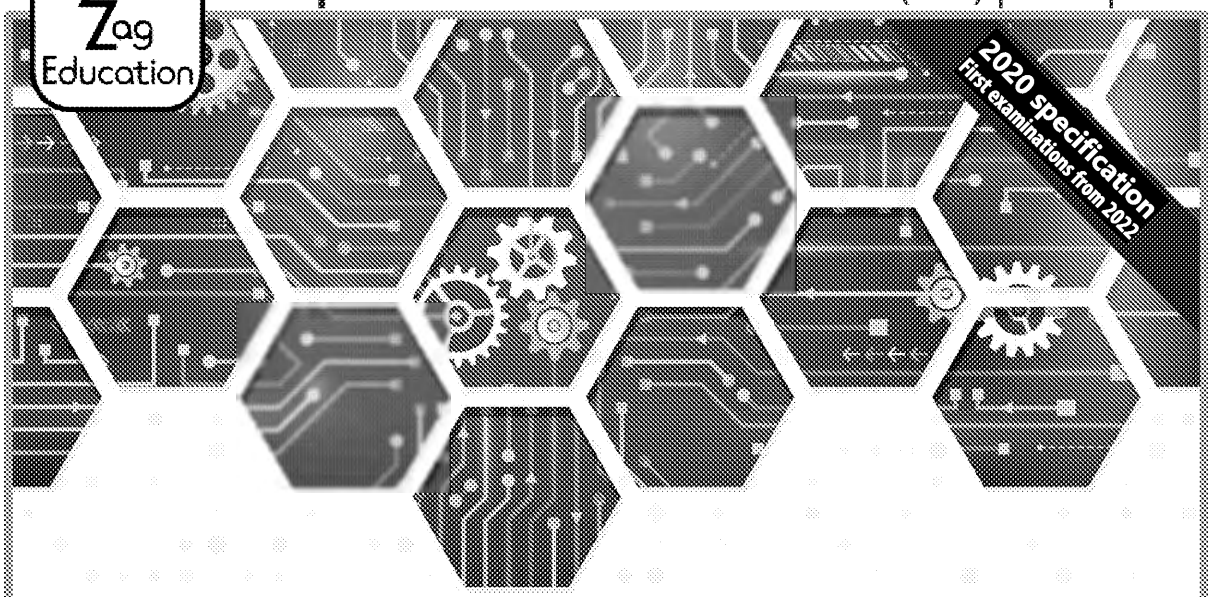
Zig Zag Education

Computer Science

GCSE (9–1) | OCR | J277



2020 specification
First examinations from 2022

PowerPoints and Worksheets for *Technical Topics*

for OCR GCSE Computer Science (J277)

A Hadwen-Bennett

DE6/
10595

POD
10595

zigzageducation.co.uk

Contents

All of the printed materials in this pack are also provided electronically on the accompanying CD

Teacher's Introduction

This resource is designed to support the delivery of the logical and mathematical concepts from the OCR GCSE (J277) specification – for first teaching in September 2020; first exams from 2022.

The topics covered are as follows:

- Data Storage Units
- Number Representation
- Binary Arithmetic
- Characters and Images
- Sound
- Computational Thinking
- Designing and Writing Programs
- Tracing Algorithms
- Sorting Algorithms
- Searching Algorithms
- Programming Concepts
- Data Types
- String Manipulation
- Data Structures & File Handling
- Sub Programs
- SQL
- Defensive Design
- Testing Programs
- Boolean Logic

For each of the topics above, there is an animated presentation, providing a step-by-step walk-through of the key concept, plus a worksheet giving students the opportunity to demonstrate their understanding.

These presentations and accompanying worksheets can be used in a number of ways:

- ✓ The animated presentations and worksheets can be used in class to introduce topics.
- ✓ The worksheets can be used as homeworks to test understanding.
- ✓ The animated presentations make perfect revision aids.
- ✓ As part of a flipped classroom, where students watch the animated presentations as preparation for the lesson. The students could complete the worksheets in class to test their understanding prior to a more in-depth discussion of the topic.

The animated presentations are provided in PowerPoint (PPTX), HTML5 and PDF formats. The HTML5 versions are included so that students can use the presentations more easily on devices which lack PowerPoint support (such as tablet computers and even smartphones), making them great for revision. Hard copies of the PDF versions have been included as an appendix at the back of this pack.

Answers for each worksheet are provided on paper and on the CD (PDF password = **10595**).

In addition to the presentations and worksheets, there is also interactive practice for the following concepts:

- Binary Conversion
- Binary Addition
- Binary Search
- Binary Shift
- Bitmap Images
- Bubble Sort
- Calculating File Sizes
- Hexadecimal
- Insertion Sort
- Logic Diagrams
- Merge Sort
- Truth Tables
- Unit Conversion

The CD contains the resource contents in a range of electronic formats, all linked together via a HTML frontend (**index.html**). If using on a network, it is recommended that you provide a shortcut to the frontend to allow easy access for students.

Alternatively, files can be accessed directly by navigating to the relevant folder on the CD.

Alex Hadwen-Bennett, September 2020

Data Storage Units

1. Give the file size shown below in bits (show your working). (2)

   2KB

2. Give the file size shown below in bits (show your working). (2)

   1MB

3. Give the file size shown below in bits (show your working). (2)

   0.5GB

4. Give the file size shown below in kilobytes (show your working). (2)

   16,000b

5. Give the file size shown below in megabytes (show your working). (2)

   800,000b

6. Give the file size shown below in megabytes (show your working). (2)

   50,000KB

7. Give the file size shown below in gigabytes (show your working). (2)

   2,000MB

8. Calculate the storage requirements for the text file detailed below. (2)

Number of Characters: 1000, Bits Per Character: 8

9. Calculate the storage requirements for the text file detailed below. Give

Number of Characters: 100, Bits Per Character: 8

10. Calculate the storage requirements for the text shown below (using 8 bi

Lorem ipsum sit amet, consectetur adipiscing elit.

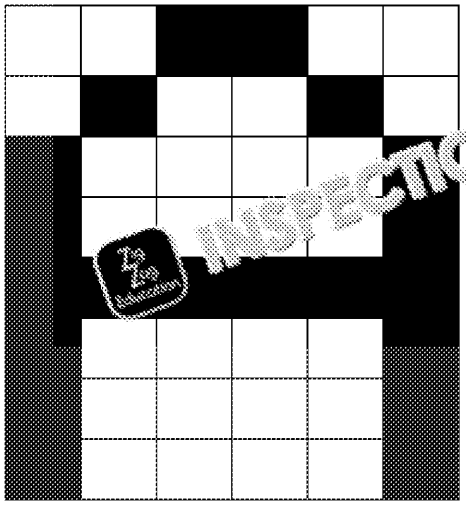
11. Calculate the storage requirements in bits for image detailed below. (2)

Width: 10, Height: 5, Colour Depth: 10

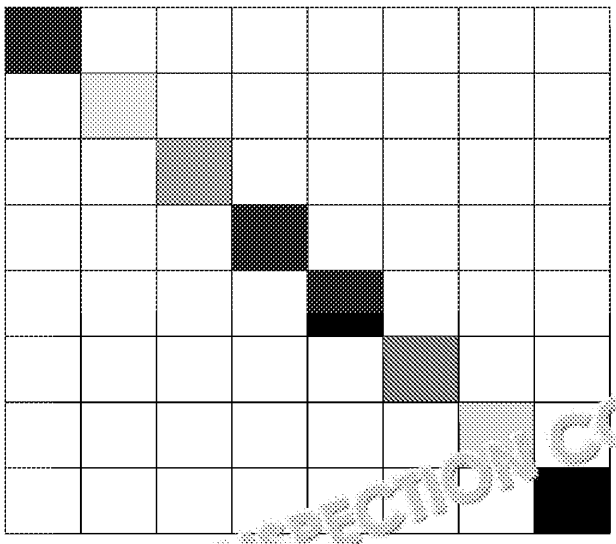
12. Calculate the storage requirements in bits for image detailed below. Giv

Width: 80, Height: 10, Colour Depth: 2

13. Calculate the storage requirements in bits for the image shown below. (



14. Calculate the storage requirements in bits for the image shown below.



15. Calculate the storage requirements in bits for the sound file detailed be

Bit Depth: 16b, Sample Rate: 1000Hz, Duration: 10 seconds

16. Calculate the storage requirements in bits for the sound file detailed be

Bit Depth: 100b, Sample Rate: 14,000Hz, Duration: 10 seconds

17. Calculate the storage requirements in bits for the sound file detailed be

Bit Depth: 24b, Sample Rate: 1000Hz, Duration: 100 seconds

18. Calculate the storage requirements in bits for the sound file detailed be kilobytes

Bit Depth: 0b, Sample Rate: 1000Hz, Duration: 6 seconds

Number Representat

1. Convert this binary number to denary. (1)

128	64	32	16	8	4	2	1
0	1	0	0	0	0	0	0

2. Convert this binary number to denary. (1)

128	64	32	16	8	4	2	1
0	0	0	0	0	[illegible]	1	1

3. Convert this binary number to denary. (1)

128	64	32	16	8	4	2	1
0	[illegible]	0	1	0	0	0	0

4. Convert this binary number to denary. (1)

128	64	32	16	8	4	2	1
0	0	0	1	0	1	1	0

5. Convert this binary number to denary. (1)

128	64	32	16	8	4	2	1
1	1	0	0	0	1	0	0

6. Convert the denary number 20 into binary. (1)

128	64	32	16	8	4	2	1

7. Convert the denary number 192 into binary. (1)

128	64	32	16	8	4	2	1

8. Convert the denary number 68 into binary. (1)

128	64	32	16	8	4	2	1

9. Convert the denary number 15 into binary. (1)

128	64	32	16	8	4	2	1

10. Convert the denary number 255 into binary. (1)

128	64	32	16	8	4	2	1

11. Convert the binary number 110101 into denary. (2)

12. Convert the binary number 10110111 into denary. (2)

13. Convert the denary number 78 into binary. (2)

14. Convert the denary number 63 into binary. (2)

15. Convert this hexadecimal number into binary. (1)

4				B			

16. Convert this hexadecimal number into binary. (1)

F				A			

17. Convert this binary number into hexadecimal. (1)

0	1	1	1	1	1	1	1

18. Convert this binary number into hexadecimal. (1)

0	1	0	1	1	1	0	1

19. Convert the denary number 186 into hexadecimal. (3)

20. Convert the denary number 97 into hexadecimal. (3)

21. Convert the hexadecimal number A7 into denary. (2)

22. Convert the hexadecimal number FF into denary. (2)

Binary Arithmetic

1. Complete the following calculations (show your working). (18)

a. 0010
+ 0001

b. 0001
+ 0011

c.
+

d. 0011
+ 0011

e. 0111
+ 0011

f.
+

g. 0011
+ 0111

h. 0111
+ 1111

i.
+

2. Identify the calculations from the previous question in which an overflo
be stored. (2)

3. Carry out this calculation: 11001 + 1110110 (2)

4. Carry out this calculation: 11010000 + 10111 (2)

5. Carry out this calculation: 110110 + 11100101 (2)

6. Carry out this calculation: 10101100 + 100110 (2)

7. Identify the calculation from the questions 3 to 6 in which an overflow was stored.

8. Perform a left binary shift of 1 on this binary number and convert both the number to denary. (3)

128	64	32	16	8	4	2	1
0	0	1	0	0	0	0	0

9. Perform a left binary shift of 2 on this binary number and convert both the number to denary. (3)

128	64	32	16	8	4	2	1
0	0	0	0	1	1	0	0

10. Perform a right binary shift of 1 on this binary number and convert both to denary. (3)

128	64	32	16	8	4	2	1
0	1	[illegible]	1	0	1	0	0

11. Perform a right binary shift of 1 to this binary number and convert both the number to denary. (3)

128	64	32	16	8	4	2	1
1	0	0	0	0	1	0	0

12. Perform a right binary shift of 1 to this binary number and convert both number to denary. (3)

128	64	32	16	8	4	2	1
1	0	0	0	0	1	0	0

13. Perform a right binary shift of 2 to this binary number and convert both number to denary. (3)

128	64	32	16	8	4	2	1
1	0	0	0	1	1	0	0

14. Perform a right binary shift of 3 to this binary number and convert both to denary. (3)

128	64	32	16	8	4	2	1
0	0	0	0	0	0	1	0

15. Perform a right binary shift of 3 to this binary number and convert both number to denary. (3)

128	64	32	16	8	4	2	1
0	0	1	0	1	0	0	1

Characters and Imag

1. Complete the character code table below: (3)

Character	Character Code (Binary)	Character Coc
F	01000110	70
G		
H		
I		

2. ASCII u or per character and Unicode uses 16 bits. What imp number racters each system can represent? (2)

3. What is the advantage of using Unicode over ASCII? (2)

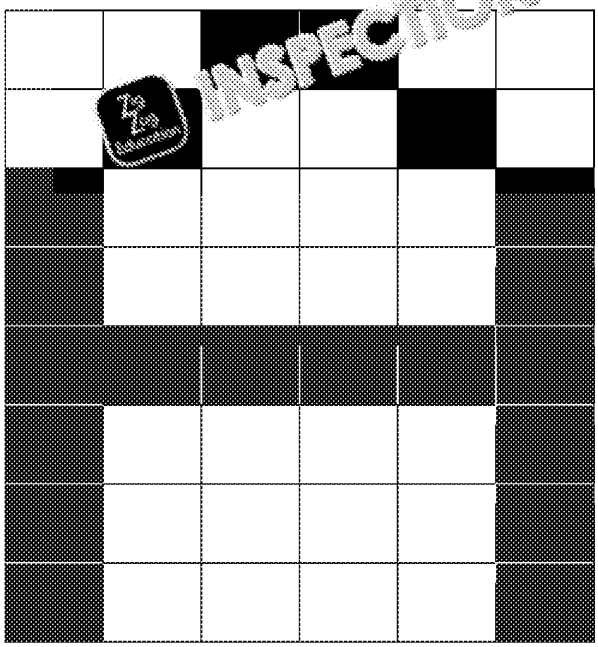
4. The bitmap image represented below uses 0 to represent white and 1 t Recreate the image from the binary code. (4)

					1,0,0,0,0
					1,0,0,0,0
					1,0,0,0,0
					1,1,1,1,0
					1,0,0,0,1
					1,0,0,0,1
					1,0,0,0,1
					1,1,1,1,0

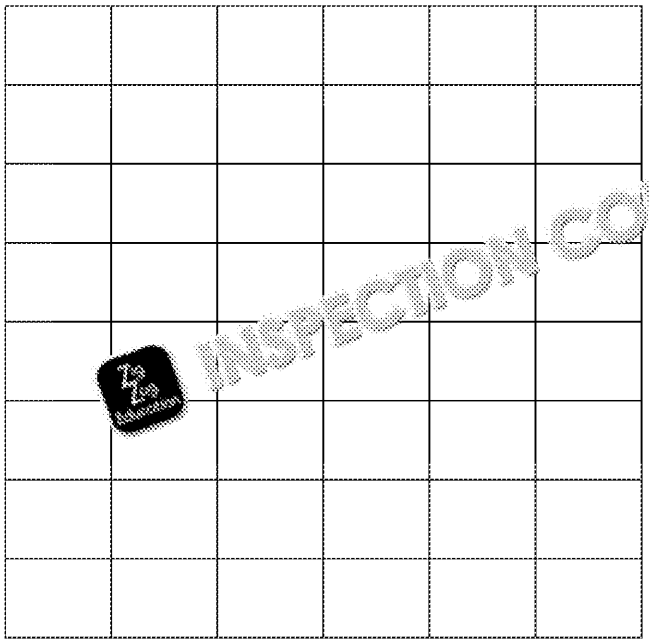
5. What is the colour depth of the image shown above? (1)

6. What is the resolution of the image shown above? (2)

7. Create the binary code to represent the image shown below. Use 0 to r represent black. (4)



8. The bitmap image represented below uses 01 to represent black, 10 to green and 00 to represent white. Recreate the image from the binary co



01, 00, 10, 00, 1

00, 01, 00, 10, 0

01, 00, 01, 00, 1

00, 01, 00, 01, 0

10, 00, 01, 00, 0

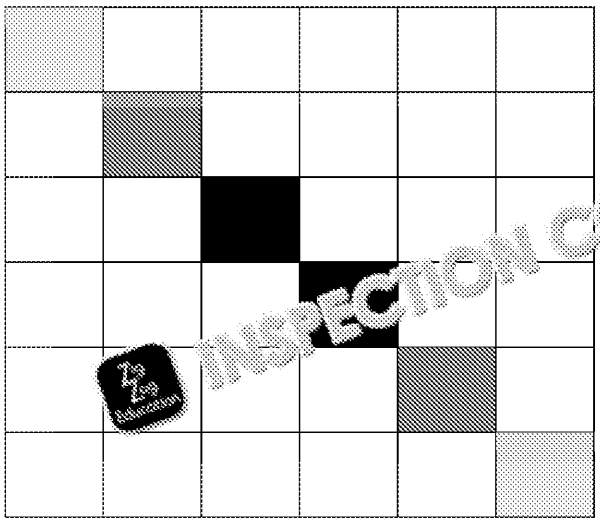
00, 10, 00, 01, 0

11, 00, 10, 00, 0

00, 11, 00, 10, 0

9. What is the colour depth of the image shown above? (1)

10. Create the binary code to represent the image shown below. Use 01 to dark grey, 11 to represent light grey and 00 to represent white. (3)



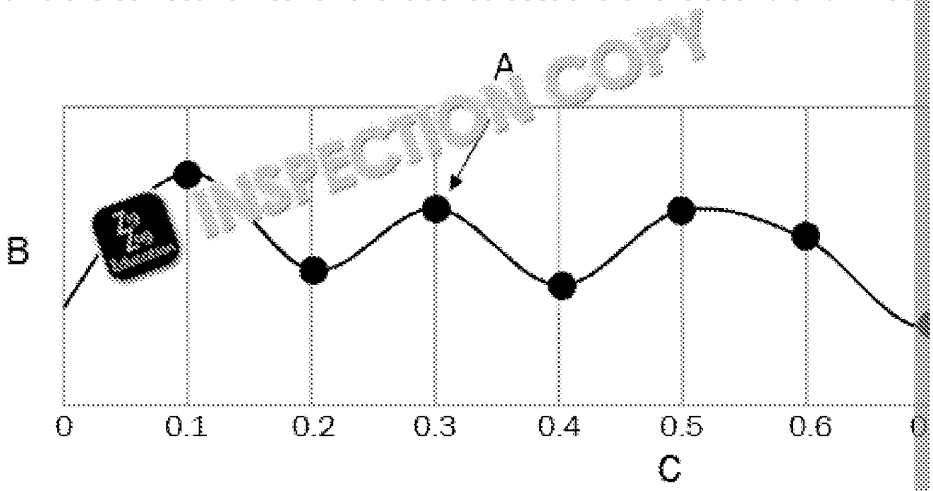
11. What is the resolution of the image shown above? (2)

12. Give an example of metadata that can be added to digital images. (1)

Sound

1. Explain why sound must be digitised to enable it to be processed by a c

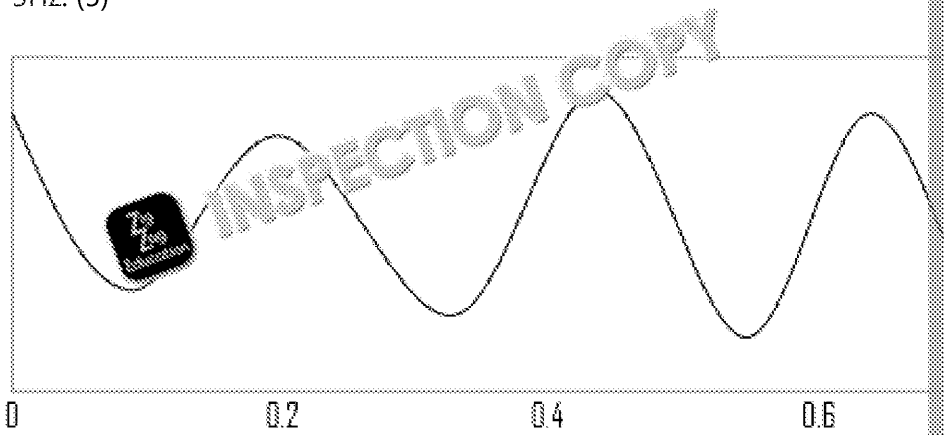
2. Give the correct names for the labelled sections of the sound shown be



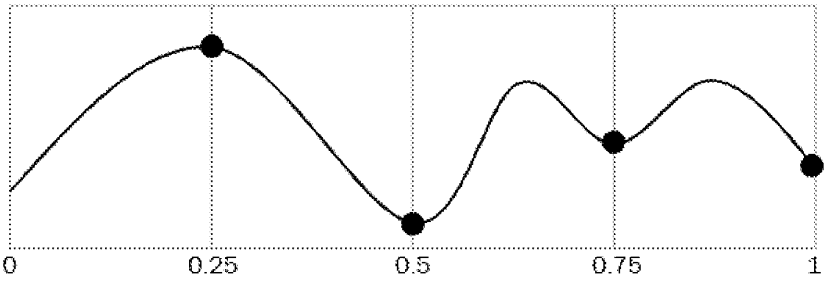
A	
B	
C	

3. What is the sample rate of the digitised sound shown above? (1)

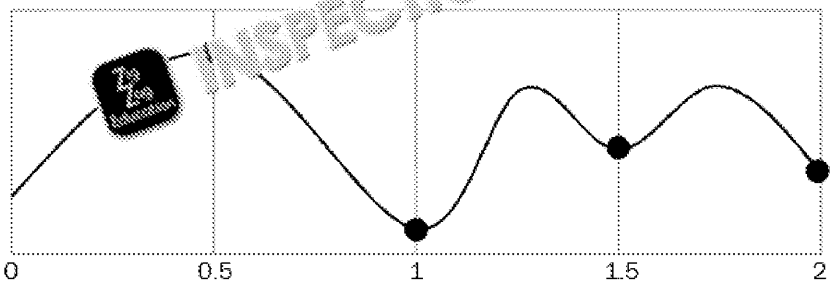
4. A sound wave is shown below, indicate at which points samples would b 5Hz. (5)



5. What is the sample rate of the digitised sound shown below? (1)



6. What is the sample rate of the digitised sound shown below? (1)



7. Describe the meaning of the term sample rate. (2)

8. Describe the meaning of the term bit depth. (2)

9. Explain the impact changing the sample rate and bit depth has on the q
the storage requirements. (3)

10. A sound has a sample rate of 20 Hz and a bit depth of 24 bits, calculate

Computational Think

Problem 1

A program is needed that will generate a sequence of numbers an
the pattern. The number sequence will start at 1 and there should b
Each number should increase by a set value, and this value should b
be selected at random each time the program runs. Here is an exa
increases by a set value of 2: 1 3 5 7 9 11 13 15 17 19. After the sequ
to the user they should be asked to guess the [illegible] number in the se
keep being offered the opportunity to guess the next number until i

1. a) *Abstract* **Pro** [illegible] y writing out the key information. (5)

b) *Decompose* **Problem 1** into its key components. (7)

Problem 2

A program is needed that will enable the user to convert between c
Celsius, Fahrenheit and Kelvin. The user should be given the option t
they wish to convert between. They will also need to be given the c
wish to have converted. After the conversion is complete the result
user. To convert between Celsius to Kelvin you add 273.15. To conv
multiply it by 9/5 and add 32 to the result. To convert Fahrenheit to
multiply the result by 5/9. To convert Fahrenheit to Kelvin you subtra
5/9 and add 273.15. To convert Kelvins to Celsius you subtract 273.1
Fahrenheit you subtract 273.15, multiply by 9/5 and add 32.

2. a) *Abstract* **Problem 2** by writing out the key information. (7)

b) *Decompose* **Problem 2** into its key components. (8)

Designing and Writing Pr

Algorithm A

```
Ask the user to input their password
While password is incorrect
     Ask them to re-enter
End while loop
Output login successful
```

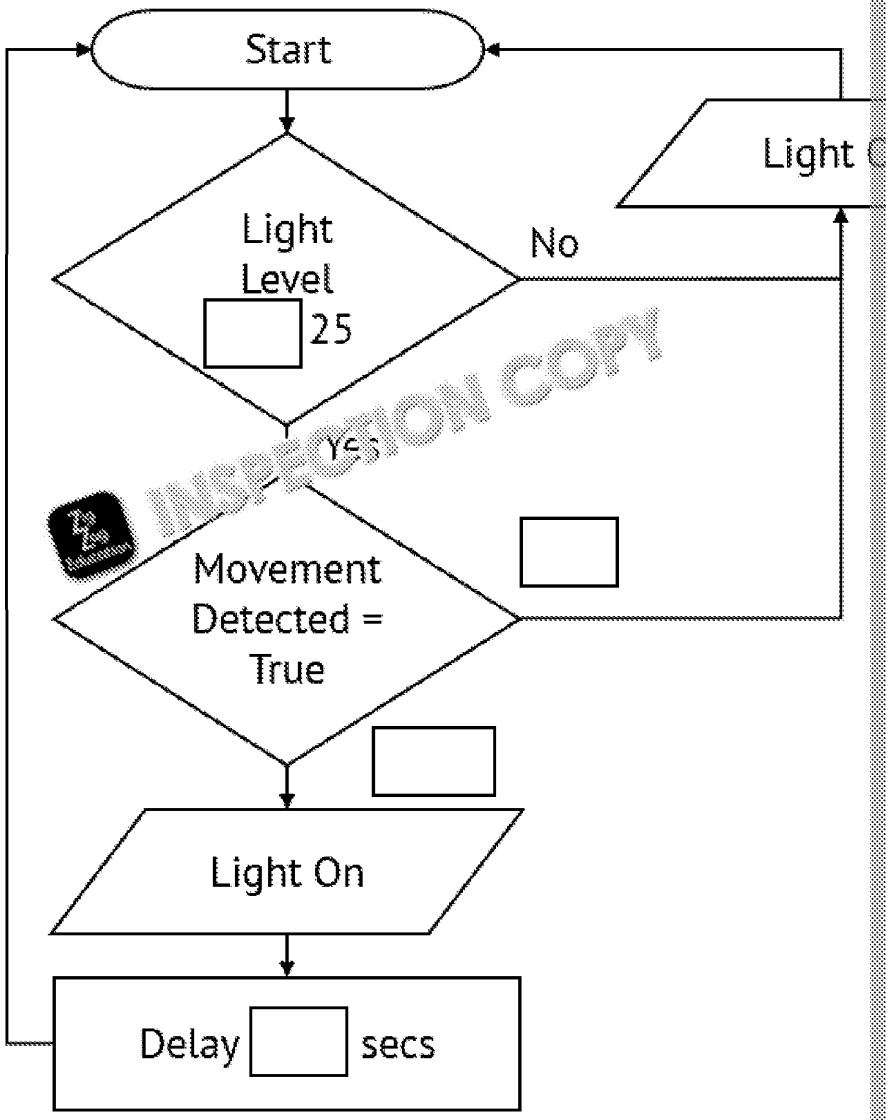
1. Which of the example algorithms are in pseudocode? (1)

2. Which of the example algorithms is in the form of a flow chart? (1)

3. Draw the flowchart symbol that matches each description. (5)

	Used to control the path taken thro the result of a condition.
	Used to indicate the start or end of
	Used to indicate a process, for exa calculation.
	Used when data needs to be input
	Used to call a pre-defined algorith

4 a) A security light is activated when it is dark and movement is detecte
below. You need to complete it. (4)



b) Design an algorithm in pseudocode based on the algorithm you con

Students are completing test. If they score 80 or above they will be given an
above they will be given a B grade, if they score 60 or above they will be giv
will be given a D grade.

5. a) Use a structure diagram to design a solution to the program given a

b) Design an algorithm in pseudocode that can be used to convert a st

5. c) Design an algorithm in the form of a flow chart that can be used to c
grade. (6)

Tracing Algorithms

1. Complete the trace table for algorithm shown below. (4)

```
start = 1
end = 4

for index = start to end
     print(index*index)
next index
```

start	end	i

2. Complete the trace table for algorithm shown below. (3)

```
countdown = 5
target = 0

while countdown >= target
     if countdown > target then
          print(countdown)
     else
          print("Blast Off!")
     endif
     countdown = countdown - 1
endwhile
```

countdown	to

3. Complete the trace table for algorithm shown below. (5)

```
array answers = [TRUE,TRUE,FALSE,FALSE,TRUE]
array responses = [TRUE,FALSE,TRUE,FALSE,TRUE]

i = 0
score = 0
len = answers.length

while i < len
     if answers[i] == responses[i] then
          score = score + 1
     endif
     i = i + 1
endwhile
```

i	score	len	answers[i]

4. Complete the trace table for algorithm shown below. (6)

```
procedure myProcedure(val1, val2, val3)
     temp1 = val1
     temp2 = val3

     val1 = temp2
     val3 = temp1

     print(val1)
     print(val2)
     print(val3)
endprocedure
myProcedure(1,2,3)
```

val1	val2	val3	temp1	temp2	Output

Sorting Algorithms

1. Complete the table below to show the state of the list after each swap w
sort algorithm. (4)

Original List	54	34	2
Swap 1			
Swap 2			
Swap 3			
Swap 4			

2. Complete the table below to show the state of the list after each full sw
bubble sort algorithm. (3)

Original List	78	54	6
Pass 1			
Pass 2			
Pass 3			

3. Complete the table below to show the state of the list after each swap w
sort algorithm. (4)

Original List	Banana	Kiwi	Pear	
Swap 1				
Swap 2				
Swap 3				
Swap 4				

4. Complete the table below to show how this list would be sorted using t

Original List	54	34	2
Stage 1			
Stage 2			
Stage 3			

5. Complete the table below to show how this list would be sorted using t

Original List	25	60	12	19	45
Stage 1					
Stage 2					
Stage 3					
Stage 4					

6. Complete the table below to show how this list would be sorted using t

Original List	Banana	Kiwi	Pear	Apple	Orange
Stage 1					
Stage 2					
Stage 3					
Stage 4					

7. Complete the table below to show how this list would be sorted using t

25, 60, 12, 19, 45, 32

	Sorted	Unsorted
Stage 1		
Stage		
Stage 3		
Stage 4		
Stage 5		
Stage 6		
Stage 7		

8. Complete the table below to show how this list would be sorted using t

Banana, Kiwi, Pear, Apple, Grape, Peach

	Sorted	Unsorted
Stage 1		
Stage 2		
Stage 3		
Stage 4		
Stage 5		
Stage 6		
Stage 7		

9. Complete the table below to show how this list would be sorted using t

25, 60, 12, 19, 45, 32, 79, 27

	Sorted	Unsorted
Stage 1		
Stage 2		
Stage 3		
Stage 4		
Stage 5		
Stage 6		
Stage 7		
Stage		
Stage		

10. Identify these two standard algorithms. (2)

Code

```
for i=1 to list.length - 1
   pos = i
   while pos > 0 AND list[i] < list[pos-1]
     temp = list[i]
     list[pos] = list[pos+1]
     list[pos+1] = temp
     pos = pos - 1
   endwhile
next i
```

```
active = true
while active == true
   active = false
   for i=0 to list.length - 2
     if list[i] > list[i+1] then
          temp = list[i]
          list[i] = list[i+1]
          list[i+1] = temp
          active = true
     endif
   next i
endwhile
```

Searching Algorithm

List A	19, 57, 44, 12, 31, 6, 98, 45
List B	18, 24, 31, 44, 58, 63, 70, 98
List C	2, 4, 7, 10, 15, 17, 21, 27
List D	65, 78, 33, 81, 20, 5, 23, 56
List E	Apple, Banana, Kiwi, Orange, Pear
List F	Banana, Kiwi, Pear, Apple, Grape, Peach
Lift G	Apple, Banana, Grape, Kiwi, Orange, Pear, Peach

1. Identify four lists that could be searched using the binary search algorit

2. State the reason other lists could not be searched using the binary sear

3. Name the algorithm that could be used to search the lists that you did n

4. How many comparisons would be needed to find the value 5 in the List identified in question 3? (1)

5. How many comparisons would be needed to find 'Pear' in the List E usi in question 3? (1)

6. Complete the table below to show how the binary search algorithm cou value 3 . The first stage has been completed for you. (2)

18	24	31	44	58	6
			↑		

7. Complete the table below to show how the binary search algorithm cou
value 'Peach' in List E. (3)

8. Complete the table below to show how the binary search algorithm cou
value 27 in List C. (4)

9. Identify these two standard algorithms. (2)

Code

```
function searchA(list, target)
   first = 0
   last = list.length
   while first != last
     mid = (first + last) DIV 2
     if target == list[mid] then
          return mid
     else if list[mid]< target then
          last = mid - 1
     else
          first = mid + 1
     endif
   endwhile
   return -1
endfunction
```

```
function searchB(list, target)
   i = 0
   while index < list.length
     if list[i] == target then
          return i
     endif
     index = index + 1
   endwhile
   return -1
endfunction
```

Programming Conce

Program A

```
01 repeat = true
02 while repeat == true
03       num1 = input()
04       num2 = input()
05       total = num1 + num2
06       print(total)
07       choice = input()
08       if choice == "N" then
09             repeat = false
10       endif
11 endwhile
```

Program B

```
num = input()
for index = 1 to num
      print("Hello")
next index
```

Pro

```
con
ite
tot
for

nex
dis
new
pri
```


1. Give th[illegible]ers of a sequence in Program A. (1)

2. Give the line number(s) that feature user inputs in each program. (3)

Program A	
Program B	
Program C	

3. Give the line number(s) that feature outputs in each program. (3)

Program A	
Program B	
Program C	

4. Name a variable from each program. (3)

Program A	
Program B	
Program C	

5. Identify the name of a constant and the program it appears in. (2)

Constant name		Program

6. Give the line number where iteration occurs in each program. (3)

Program A	
Program B	
Program C	

7. Identify the program that features a selection statement and give the li

Program		Line Number

8. Which program(s) feature a count-controlled loop? (1)

9. Which program(s) feature a condition-controlled loop? (1)

10. Describe the purpose of each program. (6)

Program A	
Program B	
Program C	

11. Identify the comparison operators. (3)

==	
>	
<	
>=	
<=	
!=	

12. Identify the output of Program D below given the following inputs. (3)

Program D

```
num = input()
if num MOD 2 == 0 AND num > 0 th
    print("Type 1")
elseif num MOD 2 > 0 then
    print("Type 2")
else
    print("Invalid")
endif
```

0	
10	
3	

Data Types

1. Identify the appropriate data type for each variable. (3)

Variable Name	Example Data	D
PlayerName	"John Smith"	
BestScore	5000	
WorstScore	105	
AverageScore	3437.5	
GameC[illegible]eted	False	
Difficulty	"M"	

2. Explain why the data type you chose for AverageScore is suitable. (1

3. Identify the appropriate data type for each variable. (3)

Variable Name	Example Data	D
BookName	"Great Expectations"	
InStock?	True	
Price	5.99	
Pages	544	
YearPublished	1861	
Auth[illegible]	"Charles Dickens"	

4. Give a suitable example and identify the appropriate data type for each

Variable Name	Example Data	D
FirstName		
LastName		
Gender		
Age		
AverageAttendance		
Curre St ?		

Example Program	
01	numA = "62"
02	numB = "54"
03	numC = "32"
04	numA_i = int(numA)
05	numB_i = int(numB)
06	numC_i = int(numC)
07	total = numA_i + numB_i + num
08	total_s = str(total)
09	print("Total: " + total_s)

5. State the data types of the following variables from the example progra

numA	
numA_i	
total_s	

6. Give the nu in which casting occurs in the example program.

7. State the output of the example program. (1)

String Manipulatio

```
#variables:
message1 = "I love "
message2 = "Computer Science"
message3 = "My lucky number is "
num1 = 5
num2 = "5"
```

1. What would be the output of the program shown below? (1)

```
message = message1 + message2
print(message)
```

2. What would be the output of the program shown below? (1)

```
message = message3 + str(num1)
print(message)
```

3. What would be the output of the program shown below? (1)

```
total = str(num1) + num2
print(total)
```

4. Write the code to output the message "I love Computer Science and My
use of all the variables defined at the top of the page. (3)

5. Write the code to output the length of the string stored in the `messag`

6. Write the code to output "love" from the string stored in the `message`

7. Write the code to output the string stored in the `message2` variable in

8. Write the code to output the string stored in the `message2` variable in

9. Write the code to output 'ce' from the `message2` variable in uppercase

10. Write the code for a program that does the following: (5)

    - asks the user to input a sentence
    - works out the length of characters and outputs it in this format: 'The [correct number of characters] characters in the sentence'
    - converts the sentence to uppercase and outputs it in this format:

11. Write the code for a program that does the following: (3)

    - asks the user to input a character
    - converts the character to its ASCII code
    - outputs it in this format: 'The ASCII code for [inputted character

12. Write the code for a program that does the following: (3)

    - asks the user to input a character code
    - converts the character code to a character
    - outputs it in this format: '[character code] is the character code

Data Structures & File H

Index	0	1	2	3
Value	Ben	Susan	Polly	Steven

1. Which value would `names[2]` return from the names array shown abo

2. Write the code to access the value "Victoria" from the names array show

3. Write the code to create the names array shown above. (2)

Index	0	1	2	3
Value	87	16	58	29

4. Which value would `sales[1]` return from the sales array shown above

5. Write the code to access the value 29 from the sales array shown above

6. Write the code to create the sales array shown above. (2)

7. Write the code to loop through and output each value in the names ar

Index	0	1	2	3
0	98	34	83	19
1	12	25	70	63
2	38	32	54	81

8. Which value would `results[1,2]` return from the results array show

9. Write the code to access the value 81 from the re array shown abo

10. Write th le te the results array shown above. (3)

11. Draw a table to represent the array that this code would create. (4)

```
array ages = [[23,65,12,76], [41,76,34,87], [61,54,6
```

12. Write a program that reads the data from the **visitors.txt** file, calculates back to the file. (5)

visitors.txt

98
67
3
79
23

Sub Programs

Program A	Program B
01 Num1 = input("Enter Num 1")	Num1 = input("E
02 Num2 = input("Enter Num 2")	Num2 = input("E
03 Num3 = input("Enter Num 3")	Num3 = input("E
04 Total = 0	
05	[redacted] Addit
06 [redacted] Addition(A,B,C)	Total = Nu
07 SubTotal = A + B + C	return Tot
08 end[redacted]	end[redacted]
09	
10 Addition(Num1,Num2,Num3)	Answer = Additi
11	
12 print(Total)	print(Answer)

Note: the sub program type has been blacked out to avoid giving

1. Which one of the two programs shown above features a function? How

2. Identify the parameters that are used in **Program A**. (3)

3. Identify the parameters that are used in **Program B**. (3)

4. Give the line number where a global variable is declared in **Program A**.

5. Give the line number where a local variable is declared in **Program A**. (1)

6. Name a global variable from **Program B**. (1)

7. Name a local variable from **Program B**. (1)

8. Given the inputs 3, 4, 3 what would be output of **Program A** be? (1)

9. Given the inputs 3, 4, 3 what would be output of **Program B** be? (1)

10. Design a procedure that takes two numbers, multiplies the first by the se

11. Design a function th[illegible] two numbers, divides the first by the secon

12. Design a function that takes an array of integers, calculates the average

13. Write a program that u[illegible] the function you wrote in the previous quest[illegible] average of each [illegible] 1D arrays contained within the 2D temps array s[illegible]

```
array [illegible]s = [[19,22,20,17,21,18,16], [23,22,22,
                      [24,23,24,24,22,20,21], [22,20,21,
```

Structured Query Langua

Dog_ID	Name	Breed	Ag
1	Daisy	Poodle	2
2	Jack	Labrador	3
3	Max	Beagle	1

1. Give the results of the query shown below. (2)

```
SELECT Name, Breed, Age, Gender
FROM Dog
WHERE Age > 1
```

2. Write a query that returns all the male dogs, showing only the Name an

3. Write a query that returns all Labradors displayed in ascending order by

City_ID	Name	Population
LON	London	8907918
BRM	Birmingham	1153717
GLS	Glasgow	612040
LIV	Liverpool	579256

4. Give the results of the query shown below. (2)

```
SELECT Name, Population, Region
FROM City
WHERE Population < 1000000
```

5. Write a query that returns all the cities with a population larger than 70
and `Population` fields. (3)

6. Write a query that returns all cities with a population greater than 600,0
order by `Name`. Show all fields. (4)

7. Shown b is the design for a table called `DVD`. Write a query that re
of 0. Show all fields. (3)D

```
DVD (DVD_ID, Title, Rating, Genre, Stock_Lev
```

8. Write a query that returns all DVDs in the family genre, with a U rating.

9. Write a query that returns all DVDs in the action or horror genres. Show

10. Write a query that returns all DVDs with a title that starts with "The". Sh

11. Write a query that returns all DVDs in the horror genre with a stock leve
Only the `Title` and `Stock_Level` fields should be shown. (3)

Defensive Design

Program A

```
mobile = input()
if mobile.length != 11 then
     print("Number Invalid")
else
     print("Number Valid")
endif
```

Program B

```
age = input
if age >= 1
     print
else
     print
endif
```

Program C

```
a1 = input("Input the first value: ")
a2 = input("Input the second value: ")
c = input("A: Add, S: Subtract, M: Multiply")
if c == "A" then
r = a1 + a2
print("Result: "+str(r))
elseif c == "S" then
r = a1 - a2
print("Result: "+str(r))
else
r = a1 * a2
print("Result: "+str(r))
endif
```

1. Which of the example programs is an example of range check validatio

2. Which of the example programs is an example of length check validatio

3. A weather station records daily temperature readings in London, no rea
or below -20°C. Design a validation algorithm to check the temperature

4. Write an authentication routine that will continue to ask the user to ent
until their credentials are entered correctly. Ensure appropriate messag
and invalid credentials. Ensure the program is maintainable by using go

5. Identify three ways in which Program C can be made more maintainabl

1	
2	
3	

6. Rewrite Program C making the changes you suggested in question 5 an validation routine. (6)

Testing Programs

Program A

```
mobile = input()
if mobile.length != 11 then
    print("Number Invalid")
else
    print("Number Valid")
endif
```

Program B

```
age = input
if age >= 1
    print("
else
    print("
endif
```

1. Complete this test table for **Program A**. (4)

Description	Test Type	Test Data
Test case longer than the expected length.	Invalid	070000000000000
	Normal	

2. Complete this test table for **Program B**. (4)

Description	Test Type	Test Data
	Boundary	
	Invalid	
	Erroneous	
	Normal	

Program C

```
01 Name = input("Enter your name")
02 Age = input("Enter your age")
03 Gender = input("Enter your gender")
04
05 if Gender == "Female" then
06      Message = Name + " is my friend and he is " +
07 else
08      Message = Name + " is my friend and she is " +
09 endif
10
11 rint(Message)
```


3. Give the line number where a syntax error occurs in **Program C**. (1)

4. Correct the syntax error you identified in your answer to the previous qu

5. Give the line number where a logic error occurs in **Program C**. (1)

6. Correct the logic error you identified in your answer to the previous que

Program D

```
01 Name = input("Enter your cat's name")
02 CatAge = input("Enter your cat's age")
03
04 function AgeConvert(CatAge)
05       NewAge = CatAge * 7
06       return CatAge
07 endfunction
08
09 HumanAge = AgeConvert(CatAge)
10
11 print(Name & " is " HumanAge & in human years.)
```

7. Give the line numbers where syntax errors occur in the **Program D**. (2)

8. Correct the syntax errors you identified in your answer to the previous q

9. Give the line number where a logic error occurs in the **Program D**. (1)

10. Correct the logic error you identified in your answer to the previous que

Boolean Logic

1. Complete the table below by drawing the logic gate symbols. (3)

Gate	Symbol
NOT	
AND	
OR	

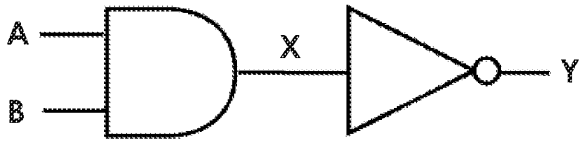
2. Complete the truth table below for a NOT gate. (2)

INPUT	OUTPUT

3. Complete the truth table below for an OR gate. (3)

OR		
INPUT 1	INPUT 2	OUTPUT

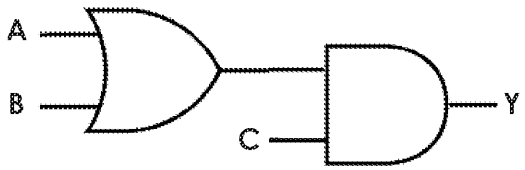
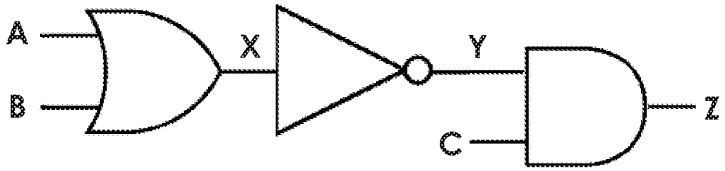
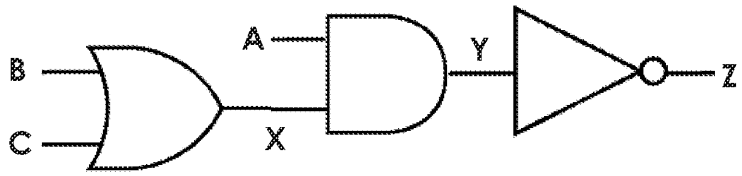
4. Complete the truth table for the logic diagram shown below. (3)



A	B	X	Y

5. Identify the logic diagram that matches the logic statement shown below.

NOT(A AND (B OR C))

NOT(A AND (B OR C))	Tick
	
	
	

6. Complete the truth table for the logic diagram you identified in the previous question.

A	B	C		

7. Construct a logic diagram to represent the logic of the scenario below. Us

- A greenhouse climate control system to control the opening of a ven
- A temperature (T) sensor is used to monitor the temperature inside t
- A humidity (H) is used to monitor the humidity inside the greenhous
- A manual switch (M) is used to manually open the vent.
- The vent (V) is opened is either the temperature (T) or the humidity (
  manual switch (M) has been turned on.

8. Construct a logic diagram and logic statement to represent the logic of th
symbols W, M, S and A. (4)

- A house has a security system fitted.
- A window (W) is used to detect if any of the windows have been ope
- A movement sensor (M) is used to detect if there is any movement i
- The activation switch (S) is used to turn the alarm system on.
- The alarm (A) is sounded if the alarm system is on and either the one
  opened or movement (M) is detected inside the house.

9. Create a truth table for the logic diagram you created in question 7. (5)

Suggested Answer

Data Storage Units

1. 1 mark for correct answer (16,000b) 1 mark for working (e.g. 2 * 1000 * 8)
2. 1 mark for correct answer (8,000,000b) 1 mark for working (e.g. 1 * 1000 * 100
3. 1 mark for correct answer (500,000,000b) 1 mark for working (e.g. 0.5 * 1000 *
4. 1 mark for correct answer (2KB) 1 mark for working (e.g. 16,000 / 8 / 1000)
5. 1 mark for correct answer (0.1MB) 1 mark for working (e.g. 800,000 / 8 / 1000
6. 1 mark for correct answer (50MB) 1 mark for working (e.g. 400,000 / 1000)
7. 1 mark for correct answer (2GB) 1 mark for working (e.g. 2,000 / 1000)
8. 1 mark for correct answer (8000b) 1 mark for working (e.g. 1000 x 8)
9. 1 mark for correct answer in bits (800b) 1 mark for working (e.g. 100 x 8) 1 ma
10. 1 mark for correct answer (400b) 1 mark for working (e.g. 50 x 8)
11. 1 mark for correct answer (500b) 1 mark for working (e.g. 10 x 5 x 10)
12. 1 mark for correct answer in bits (1600b) 1 mark for working (e.g. 80 x 10 x 2) 1 (200B)
13. 1 mark for correct answer (48b) 1 mark for working (e.g. 6 x 8 x 1)
14. 1 mark for correct answer (128b) 1 mark for working (e.g. 8 x 8 x 2)
15. 1 mark for correct answer (160,000b) 1 mark for working (e.g. 16 x 1000 x 10)
16. 1 mark for correct answer (1,400,000b) 1 mark for working (e.g. 100 x 14,000 x
17. 1 mark for correct answer (2,400,000b) 1 mark for working (e.g. 24 x 1000 x 1
18. 1 mark for correct answer in bits (600,000b) 1 mark for working (e.g. 100 x 100 kilobytes (75KB)

Number Representation

1. 64
2. 7
3. 80
4. 22
5. 196
6. 0001 0100
7. 1100 0000
8. 0100 0100
9. 0000 1111
10. 1111 1111
11. 1 mark for working (32 + 16 + [illegible]) and 1 mark for the correct answer (53)
12. 1 mark for working ([illegible] + 16 + 4 + 2 + 1) and 1 mark for the correct answ
13. 1 mark for working and 1 mark for the correct answer (01001110)

128	64	32	16	8	4	2	1
0	1	0	0	1	1	1	0

14. 1 mark for working and 1 mark for the correct answer (00111111)

128	64	32	16	8	4	2	1
0	0	1	1	1	1	1	1

15. 0100 1011
16. 1111 1010
17. 6F

18. 5D
19. 1 mark for working (e.g. converting to binary first – 1011 1011) and 1 mark for
20. 1 mark for working (e.g. converting to binary first – 0110 0001) and 1 mark for
21. 1 mark for working (e.g. converting to binary first - 1010 0111) and 1 mark for
22. 1 mark for working (e.g. converting to binary first – 1111 1111) and 1 mark for

Binary Arithmetic

1. 2 marks available for each calculation, 1 for the correct answer and 1 for the w

a.
```
  0010
+ 0001
------
  0011
```

b.
```
  0001
+ 0011
------
  0100
```

c.
```
  0011
+ 0100
------
  0111
```

d.
```
   11
  0011
+ 001
------
  0110
```

e.
```
  0111
+ 0011
------
  1010
```

f.
```
   1
  0101
+ 0110
------
  1011
```

g.
```
  111
  0011
+ 0111
------
  1010
```

h.
```
  111
  0111
+ 1111
------
  10110
```

i.
```
  111
  1111
+ 1111
------
  11110
```

2. h and i.

3. 1 mark for the correct answer and 1 for the working. (2)

```
  111
  00011001
+ 01110110
----------
  10001111
```

4. 1 mark for the correct answer and 1 for the working. (2)

```
   1
  11010000
+ 00010111
----------
  11100111
```

5. 1 mark for the correct answer and 1 for the working. (2)

```
  11  1
  00110110
+ 11100101
----------
  100011011
```

6. 1 mark i correct answer and 1 for the working. (2)

```
  1 11
  10101100
+ 00100110
----------
  11010010
```

7. Question 5

8. 1 mark for the correct shifted value, 1 mark for the correct conversion of the orig
mark for the correct conversion of the new number to decimal.

128	64	32	16	8	4	2	1
0	0	1	0	0	0	0	0
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**

9. 1 mark for the correct shifted value, 1 mark for the correct conversion of the orig
mark for the correct conversion of the new number to decimal.

128	64	32	16	8	4	2	1
0	0	0	0	1	1	0	0
0	**0**	**1**	**1**	**0**	**0**	**0**	**0**

10. 1 mark for the correct shifted value, 1 mark for the correct conversion of the orig
mark for the correct conversion of the new number to decimal.

128	64	32	16	8	4	2	1
0	1	[illegible]	1	0	1	0	0
0	**0**	**1**	**0**	**1**	**0**	**1**	**0**

11. 1 mark for the correct shifted value, 1 mark for the correct conversion of the orig
mark for the correct conversion of the new number to decimal.

128	64	32	16	8	4	2	1
1	0	0	0	0	1	0	0
0	**1**	**0**	**0**	**0**	**0**	**1**	**0**

12. 1 mark for the correct shifted value, 1 mark for the correct conversion of the orig
mark for the correct conversion of the new number to decimal.

128	64	32	16	8	4	2	1
1	0	0	0	1	1	0	0
0	**0**	**1**	**0**	**0**	**0**	**1**	**1**

13. 1 mark for the correct shifted value, 1 mark for the correct conversion of the orig
mark for the correct conversion of the new number to decimal.

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1
0	**0**	**0**	**1**	**1**	**1**	**1**	**1**

14. 1 mark for the correct shifted value, 1 mark for the correct conversion of the orig
mark for the correct conversion of the new number to decimal.

128	64	32	16	8	4	2	1
0	0	0	0	0	[illegible]	1	0
0	**0**	**0**	**1**	[illegible]	**0**	**0**	**0**

15. 1 mark for the correct shifted value, 1 mark for the correct conversion of the orig
mark for the correct conversion of the new number to decimal.

128	64	32	16	8	4	2	1
0	0	1	0	1	0	0	1
0	**0**	**0**	**0**	**0**	**1**	**0**	**1**

Characters and Images

1. 1 mark per correct row (3 marks):

Character	Character Code (Binary)	Character Code (D
F	01000110	70
G	*01000111*	*71*
H	*01001000*	*72*
I	*01001001*	*73*

2. Unicode has more possible character codes (1) and can therefore represent m
3. ASCII can only represent enough characters to represent the symbols used in can represent enough characters to represent all symbols used in all language
4. 1 mark for each pair of correct rows:

1	0	0	0	0
1	0	0	0	[illegible]
1	[illegible]	[illegible]	[illegible]	0
1	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	0

5. 1 bit
6. 5 x 8 (1) = 40 (1)
7. 1 mark for each pair of correct rows:

0	0	1	1	0	0	0,0,1,1,0,0
0	1	0	0	1	0	0,1,0,0,1,0
1	0	0	0	0	1	1,0,0,0,0,1
1	0	0	0	0	1	1,0,0,0,0,1
1	1	1	1	1	1	1,1,1,1,1,1
1	0	0	0	0	1	1,0,0,0,0,1
1	0	0	0	0	1	1,0,0,0,0,1
1	0	0	0	0	1	1,0,0,0,0,1

8. 1 mark for each pair of correct rows:

01	00	10	00	11	[illegible]
00	01	[illegible]	[illegible]	[illegible]	11
01	[illegible]	[illegible]	00	10	00
00	01	00	01	00	10
10	00	01	00	01	00
00	10	00	01	00	01
11	00	10	00	01	00
00	11	00	10	00	01

9. 2 bits

10. 1 mark for each pair of correct rows:

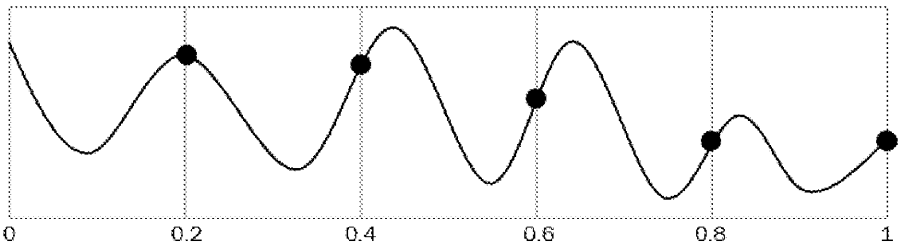
11	00	00	00	00	00	11,00,00,00,00,00
00	10	00	00	00	00	00,10,00,00,00,00
00	00	01	00	00	00	00,00,01,00,00,00
00	00	00	01	00	00	00,00,00,01,00,00
00	00	00	00	10	00	00,00,00,00,10,00
00	00	00	00	00	11	00,00,00,00,00,11

11. 6 x 6 (1) = 36 (1)

12. The location where the image was taken / Keywords that describe the image,

Sound

1. Sound is analogue (1) and computers can only process data in digital form (1)
2. A. Sample, B. Amplitude, C. Time
3. 10Hz
4. 1 mark for each correctly positioned sample



5. 4Hz
6. 2Hz
7. How often the samples are taken (1) measured in hertz/Hz (1)
8. The amount of storage space (1) allocated to each sample (1)
9. Increasing the sampling rate and sample resolution (1) increases the quality of representation closer to the original sound (1) but also increases the storage re
10. 20 Hz x 24 bits (1) = 480 bit/s (1)

Computational Thinking

1a. 1 mark for any extract that covers one of these points, the wording does not h does not matter. (5 marks max)

- Sequence will start at 1 and there should be 10 numbers in total
- Each number should increase by a set value
- Value should be between 2 and 10 and be selected at random
- After the sequence has been displayed to the user they should be as
- User should keep being offered the opportunity to guess the next nu

1b. 1 mark for each component name that captures the essence of the ones given components are in a logical order

- Generate random value between 2 and 10
- Generate sequence of 10 numbers with value between them
- Output sequence
- Ask user to guess next value
- Check if guess is correct
- Repeat until the guess is correct

2a. 1 mark for each extract that covers one of the points below, the wording does does not matter, 1 mark for details of Celsius to Kelvins and Celsius to Fahren Fahrenheit to Celsius and Fahrenheit to Kelvins, 1 mark for details of Kelvins t Fahrenheit.

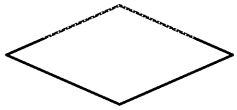
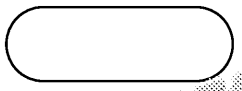
- Convert between different temperature units, Celsius, Fahrenheit and K
- Option to choose which pair of units the user wish to convert between
- Option to enter the value to be converted
- Result should be output to the user

2b. 1 mark for each component name that captures the essence of one of the rows mark if the components are in a logical order

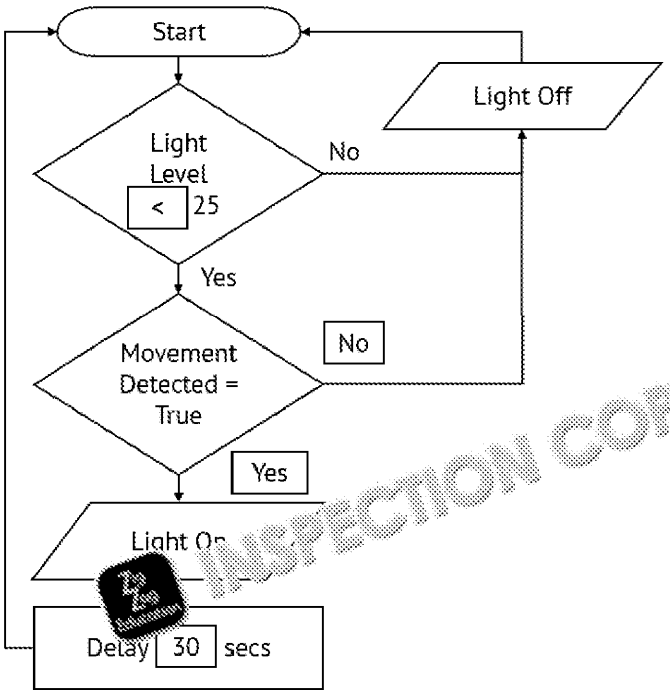
User to input value
Menu to choose between Celsius, Fahrenheit and Kelvins
Call appropriate sub program
Celsius to Kelvins sub program Celsius to Fahrenheit sub program
Fahrenheit to Celsius sub program Fahrenheit to Kelvins sub program
Kelvins to Celsius sub program Kelvins to Fahrenheit subprogram
Output to converted value

Designing and Writing Programs

1. Algorithm B
2. Algorithm A
3. 1 mark per correct symbol:

	Used to control the path taken throug based on the result of a condition.
	Used to indicate the start or end of a
	Used to indicate a process, for exam a calculation.
	Used when data needs to be inputte
	Used to call a pre-defined algorithm.

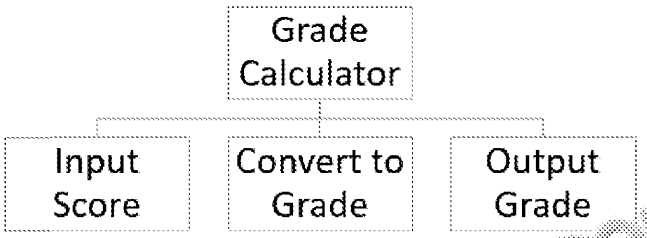
4a. 1 mark for each correctly completed box:



4b. 1 mark for putting the condition within a loop structure, 1 mark for light turned o
for light turned off if condition aren't met, 1 mark for 30 second delay before re

```
While light is active
    If the light level is less than 25 and movement is det
        Turn the light on
        Wait for 30 seconds before repeating
    Else
        Turn the light off
    End if statement
End while loop
```

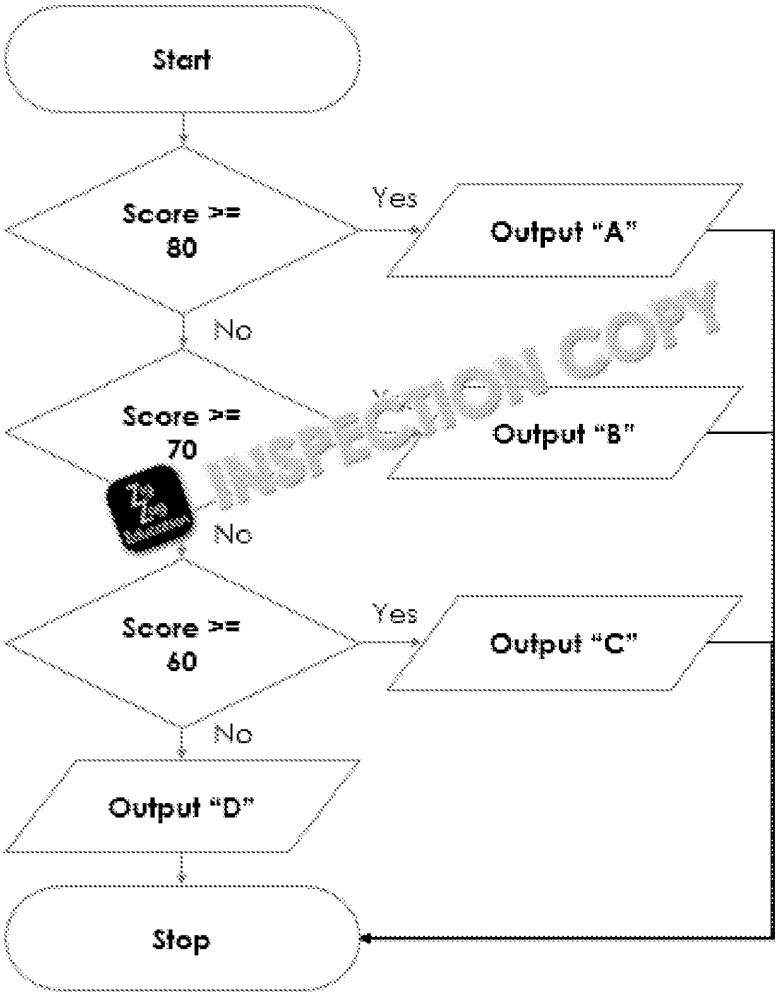
5a. *1 mark for a single top-level node with an appropriate title that describes the w*
level node that is a logical sub-task (to a maximum of 3).



5b. 1 mark for correct logic for conv of A grade, 1 mark for correct logic for c
correct logic for conversi de, 1 mark for correct logic for conversion o

```
If sco  greater than or equal to 80 then
    Out
Else if score is greater than or equal to 70 then
    Output B
Else if score is greater than or equal to 60 then
    Output C
Else
    Output D
End if statement
```

5c. 1 mark for correct logic for conversion of A grade, 1 mark for correct logic for c correct logic for conversion of C grade, 1 mark for correct logic for conversion o of input/output symbols, 1 mark for correct use of decision symbols.



Tracing Algorithms

1. 1 mark per column with values in the correct order (spacing between rows is n

Start	End	Index	Output
1	5		
		1	
			1
		2	
			4
		3	
			9
		4	
			16

2. 1 mark per column with values in the correct order (spacing between rows is n

Countdown	Target	Output
5	0	
		5
4		
		4
3		
		3
2		
		2
1		
		1
0		
		"Pas. C."
-1		

3. 1 mark p mn with values in the correct order (spacing between rows is n

i	Score	Len	Answers[i]	Responses[i]
0	0	5		
			TRUE	TRUE
	1			
1				
			TRUE	FALSE
2				
			FALSE	TRUE
3				
			FALSE	FALSE
	2			
4				
			TRUE	TRUE
	3			
5				

4. 1 mark per column with values in the correct order (spacing between rows is n

Val1	Val2	Val3	Temp1	Temp2	Output
1	2	3	1	3	
3		1			
					1

Sorting Algorithms

1. 1 mark for each correctly completed row. (4)

Original List	54	34	2
Swap 1	34	54	2
Swap 2	34	21	5
Swap 3	34	21	4
Swap 4	21	34	4

2. 1 mark for each correctly completed row. (3)

Original List	78	54	6
Pass 1	54	6	3
Pass 2	6	31	5
Pass 3	6	31	5

3. 1 mark for each correctly completed row. (4)

Original List	Banana	Kiwi	Pear	
Swap 1	Banana	Kiwi	Apple	
Swap 2	Banana	Kiwi	Apple	
Swap 3	Banana	Apple	Kiwi	
Swap 4	Apple	Banana	Kiwi	

4. 1 mark for each correctly completed row. (3)

Original List	54	34	2
Stage 1	54	34	2
Stage 2	34	54	2
Stage 3	21	34	4

5. 1 mark for each correctly completed row. (4)

Original List	25	60	12	19	45
Stage 1	25	60	12	19	45
Stage 2	25	60	12	19	32
Stage 3	12	19	25	60	28
Stage 4	12	19	25	28	32

6. 1 mark for each correctly completed row. (4)

Original List	Banana	Kiwi	Pear	Apple	Orange
Stage 1	Banana	Kiwi	Pear	Apple	Orange
Stage 2	Banana	Kiwi	Apple	Pear	Apricot
Stage 3	Apple	Banana	Kiwi	Pear	Apricot
Stage 4	Apple	Apricot	Banana	Grape	Kiwi

7. 1 mark for each pair of correctly completed rows (stages 1 to 6), 1 mark for co

Stage 1		25, 60, 12, 1
Stage 2	25	60, 12, 19, 4
Stage 3	25, 60	12, 19, 45, 3
Stage 4	12, 25, 60	19, 45, 32
Stage 5	12, 19, 25, 60	45, 32
Stage 6	12, 19, 25, 45, 60	32
Stage 7	12, 19, 25, 32, 45, 60	

8. 1 mark for each pair of correctly completed rows (stages 1 to 6), 1 mark for co

Stage 1		Banana, Kiw
Stage 2	Banana	Kiwi, Pear, A
Stage 3	Banana, Kiwi	Pear, Apple
Stage 4	Banana, Kiwi, Pear	Apple, Grap
Stage 5	Apple, Banana, Kiwi, Pear	Grape, Pea
Stage 6	Apple, Banana, Grape, Kiwi, Pear	Peach
Stage 7	Apple, Banana, Grape, Kiwi, Peach, Pear	

9. 1 mark for each pair of correctly completed rows (stages 1 to 8), 1 mark for co

	Sorted	Unsorted
Stage 1		25, 60, 12, 19, 45
Stage 2	25	60, 12, 19, 45, 32
Stage 3	25, 60	12, 19, 45, 32, 79
Stage 4	12, 25, 60	19, 45, 32, 79, 27
Stage 5	12, 19, 25, 60	45, 32, 79, 27
Stage 6	12, 19, 25, 45, 60	32, 79, 27
Stage 7	12, 19, 25, 32, 45, 60	79, 27
Stage 8	12, 19, 25, 32, 45, 60, 79	27
Stage 9	12, 19, 25, 27, 32, 45, 60, 79	

10. 1 mark for each correctly identified algorithm. (2)

Code	Algorithm

`for i=1 to list.length - 1`
`  pos = i`
`  while pos > 0 AND list[i] < list[pos-1]`
`  temp = list[i]`
`  list[pos] = list[pos+1]`
`  list[pos+1] = temp`
`  pos = pos - 1`
`  endwhile`
`next i`
	Insertion Sort

`active = true`
`while active == true`
`  active = false`
`  for i=0 to list.length - 2`
`  if list[i] > list[i+1] then`
`  temp = list[i]`
`  list[i] = list[i+1]`
`  list[i+1] = temp`
`  active = true`
`  endif`
`  next i`
`endwhile`
	Bubble Sort

Searching algorithms

1. B (1), C (1), E (1) and G (1)
2. They are not sorted/ordered
3. Linear search
4. 6
5. 5

6. 1 per correct stage and correctly positioned midpoint.

18	24	31	44	68	[illegible]
			↑		
18	**24**	**31**			
	↑				
		31			
		↑			

7. 1 mark per correct stage and correctly positioned midpoint.

Apple	**Banana**	**Grape**	**Kiwi**	**Orange**	**Pe**
			↑		
				Orange	**Pe**

8. 1 mark per correct stage and correctly positioned midpoint.

2	**4**	**7**	**10**	**15**	**1**
			↑		
				15	**1**

9. 1 mark for each correctly identified algorithm. (2)

Code	Algorithm

`function searchA(list, target)`
`  first = 0`
`  last = list.length`
`  while first != last`
`  mid = (first + last) DIV 2`
`  if target == list[mid] then`
`  return mid`
`  else if list[mid]< target then`
`  last = mid - 1`
`  else`
`  first = mid + 1`
`  endif`
`  endwhile`
`  return -1`
`endfunction`
	Binary Search

`function searchB(list, target)`
`  index = 0`
`  while index < list.length`
`  if list[i] == target then`
`  return i`
`  endif`
`  index = index + 1`
`  endwhile`
`  return -1`
`endfunction`
	Linear Search

Programming Concepts

1. Identify any 2 or more lines between 3 and 7
2. Program A: 3, 4 and 7 (1)
   Program B: 1 (1),
   Program C: 2 and 5 (1)
3. Program A: 6 (1)
   Program B: 3 (1)
   Program C: 10 (1)
4. Program A: `repeat/num1/num2/total/choice` (1)
   Program B: `num/index` (1)
   Program C: `items/total/i/price/disc/newTotal` (1)
5. `rate` (1) in Program C (1)
6. Program A: 2 (1)
   Program B: 2 (1)
   Program C: 4 (1)
7. Program 8
8. Program d C (1)
9. Program A (1)
10. Program A: Asks the user to input 2 numbers (1) adds them together (1) g the program (1) (2 points max)

    Program B: Asks the user to input a number (1) outputs 'Hello' that numbe

    Program C: Asks the user input a number of items (1) asks the user to inp them to a total (1) calculates the discount and subtracts it from total (1) (2 points max)
11. 1 mark per correct row

==	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!=	Not equal to

12. 0: Invalid (1)
    10: Type 1 (1)
    3: Type 2 (1)

Data Types

1. 1 mark per pair of correctly identified d es

Variable Name	Example Data	D
PlayerNam	"John Smith"	S
BestSc	5000	I
WorstSc	109	I
AverageScore	3437.5	R
GameCompleted	False	B
Difficulty	"M"	C

2. The number has a fractional part / a decimal place.

3. 1 mark per pair of correctly identified data types.

Variable Name	Example Data	D
BookName	"Great Expectations"	S
InStock?	True	B
Price	5.99	R
Pages	544	I
YearPublished	1861	I
Author	"Charles Dickens"	S

4. 1 mark per row with a suitable example and appropriate data type.

Variable Name	Example Data	D
FirstName	e.g. "Alex"	S
LastName	e.g. "Roberts"	S
Gender	e.g. "M" or "Male"	C
Age	e.g. 15	I
Average daily	e.g. 95.1	R
Current student?	e.g. TRUE	B

5. `numA:` String (1)
   `numA_i:` Integer (1)
   `total_s:` String (1)
6. 4, 5, 6, 8 (1)
7. 'Total: 148' (1)

String Manipulation

1. I love Computer Science (1)
2. My lucky number is 5 (1)
3. 55 (1)
4. 1 mark for concatenating at least two of the variables, 1 mark for a line of code the exact message making use of all of the variables with appropriate casting, the message to the screen.

```
message = message1 + message2 + " and " + message3 + str(
print(message)
```

5. `print(message2.length)` (1)
6. `print(message1.subString(2,4))` (1)
7. `print(message2.upper)` (1)
8. `print(message2.lower)` (1)
9. 1 mark for extracting the 'ce' substring, 1 mark for converting the message to u 1 mark for outputting the message.

```
message = message1.right(2)   /   message = message2.sub
print(message2.upper)
```

10. 1 mark for asking for user input, 1 mark for calculating the number of characte converting the sentence to uppercase and outputting it with the correct messa number of characters with the correct message, 1 mark for converting number

```
sentence = input("Please input a sentence: ")
characters = sentence.length
sentence_upper = sentence.upper
print("There are "+str(characters)+" in the sentence")
print("Uppercase: "+sentence_upper)
```

11. 1 mark for asking for user input, 1 mark for converting to character code, 1 ma
with the correct message

```
character = input("Please input a character: ")
code = ASC(character)
print("The ASCII code for "+character+" is "+code)
```

12. 1 mark for asking for user input, 1 mark for converting to character code, 1 ma
with the correct message

```
code = input("Please input a character code: ")
character = CHR(code)
print(code+" is the character code for "+character)
```

Data Structures & File Handling

1. Polly
2. `names[5]`
3. 1 mark f ng the list of values in the correct order separated by commas,
names.

```
array names = ["Ben","Susan","Polly","Steven","Jamie",
```

(Don't penalise for not placing the values in quotation marks).

4. 16
5. `sales[3]`
6. 1 mark for writing the list of values in the correct order separated by commas,

```
array sales = [87, 16, 58, 29, 93, 73]
```

7. 1 mark for correct initialisation of the for loop, 1 mark for outputting the current

```
for i = 0 to names.length-1
    print(names[i])
next i
```

8. 70
9. `results[2,3]`
10. 1 mark for each row represented as a list separated by commas, 1 mark for th
inside another

```
array results = [[98,34,83,19,32,92],[12,25,70,63,72,4
```

11. 1 mark per correct row:

23	65	12	76
41	76	34	87
61	54	66	32
12	43	15	17

12. 1 mark for ni itors.txt' file, 1 mark for iterating through each line, 1
1 mark er g read value to integer, 1 mark for writing the total to the fi

```
file = open("visitors.txt")
total = 0
while NOT file.endOfFile()
    num = file.readLine()
    total = total + int(num)
endwhile
file.writeLine(total)
file.close()
```

Sub Programs

1. Program B (1) – as it returns a value (1) on line 7
2. 1 mark per parameter (`A, B, C`)
3. 1 mark per parameter (`Num1, Num2, Num3`)
4. Any of 01 / 02 / 03 / 04
5. 07
6. `Answer`
7. `Total`
8. 0
9. 10
10. 1 mark for defining a procedure that has 2 parameters, 1 mark for multiplying the outputting the result

```
procedure Multiply(Num1, Num2)
    Result = Num1 * Num2
    print(Result)
endprocedure
```

11. 1 mark for defining a function that has 2 parameters, 1 mark for dividing the first mark for returning the result

```
function Divide(Num1, Num2)
    Result = Num1 / Num2
    return Result
endfunction
```

12. 1 mark for correctly declaring a function that accepts one parameter, 1 mark for the values in the array, 1 mark for calculating the total, 1 mark for calculating the the average

```
function average(values)
    total = 0
    for i = 0 to values.length - 1
        total = total + values[i]
    next i
    averag e = total / values.length
    return average
endfunction
```

13. 1 mark for the use of a loop that will iterate over the 2D temps array, 1 mark for previous question and passing the 1D sublists to it, 1 mark for outputting the re

```
for i = 0 to temps.length-1
 print(average(temps[i]))
next i
```

SQL

1. 1 mark per correctly identified record:
   1, Daisy, [illegible]le, [illegible]
   2, Jack, [illegible]or, 5, M
2. 1 mark per correct statement:

```
SELECT Name, Breed
FROM Dog
WHERE Gender = "M"
```

3. 1 mark per correct statement:

```
SELECT *
FROM Dog
WHERE Breed = "Labrador"
ORDER BY Name
```

4. 1 mark per correct row:

Name	Population	Region
Glasgow	612040	Scotland
Liverpool	579256	North West

1 mark per correct statement for each of the following:

5.
```
SELECT Name, Population
FROM City
WHERE Population > 700000
```

6.
```
SELECT *
FROM City
WHERE Population > 600000
ORDER BY Name
```

7.
```
SELECT *
FROM DVD
WHERE Stock_Level = 0
```

8.
```
SELECT *
FROM DVD
WHERE Genre = "Family" AND Rating = "U"
```

9.
```
SELECT *
FROM DVD
WHERE Genre = "Action" OR Genre = "Horror"
```

10.
```
SELECT *
FROM DVD
WHERE Title LIKE The%
```

11.
```
SELECT Title, Stock_Level
FROM DVD
WHERE Genre = "Horror" AND Stock_Level >= 50
```

Defensive Design

1. Program B
2. Program A
3. 1 mark for input of temperature reading, 1 mark for correct logic to test if the re suitable message outputted if temperature is in range, 1 mark for suitable mes not in range.

Example Solution 1

```
Temp = input()
if temp < -20 OR temp > 40 then
    print("Temperature out of range")
else
    print("Temperature within range")
endif
```

Example Solution 2

```
Temp = input()
if temp => -20 AND temp <= 40 then
    print("Temperature within range")
else
    print("Temperature out of range")
endif
```

4. 1 mark for assigning correct username and password value to variables, 1 ma structure to keep asking the user to enter their details until they are correct, 1 password input, 1 mark for comparing entered username and password detail appropriate messages for both valid and invalid details, 1 mark for appropriate 1 mark for meaningful identifiers

```
username = "carol"
password = "key"
valid = FALSE
while valid == FALSE
    inpUsername = input("Username: ")
    inpPassword = input("Password: ")
    if inpUsername == username AND inpPassword == passwor
        valid = TRUE
        print("Logged In")
    else
        print("Invalid, please try again.")
    endif
endwhile
```

5. Adding co en ing meaningful names/identifiers (1), Using indentati *Also all ark for suggesting adding sub programs (max mark 3)*

6. 1 mark fo using meaningful names/identifiers, 1 mark for using indentation, 1 comments, 1 mark for a validation routine that ensures the user has entered o the validation routine repeating until a valid option is chosen, 1 mark for appro are multiple ways sub programs could be used, any logical use should be awa

Example Solution 1 (without sub program):

```
//ask the user to input two values for their calculation
num1 = input("Input the first value: ")
num2 = input("Input the second value: ")

//ask the user to choose the type of calculation they wou
validate their input
valid = FALSE
while valid == FALSE
    choice = input("A: Add, S: Subtract, M: Multiply")
    if choice == "A" OR choice == "S" OR choice == "M" the
        valid == TRUE
    endif
endwhile

//carries out the chosen calculation and outputs the resu
if choice == "A" then
    result = num1 + num2
    print("Result: "+str(result))
elseif choice == "S" then
    result = num1 - num2
    print("Result: "+str(result))
else
    result = num1 * num2
    print("Result: "+str(result))
endif
```

Exampl tio 2 (with a sub program):

```
//func to produce menu and validate user input
function menu()
    valid = FALSE
    while valid == FALSE
        choice = input("A: Add, S: Subtract, M: Multiply")
        if choice == "A" OR choice == "S" OR choice == "M"
            valid == TRUE
        endif
    endwhile
    return choice
endfunction
```

```
//ask the user to input two values for their calculation
num1 = input("Input the first value: ")
num2 = input("Input the second value: ")

//ask the user to choose the type of calculation they wou
choice = menu()

//carries out the chosen calculation and outputs the resu
if choice == "A" then
    result = num1 + num2
    print("Result: "+str(result))
elseif choice == "S" then
    result = num1 - num2
    print("Result: "+str(result))
else
    result = num1 * num2
    print("Result: "+str(result
endif
```

Testing [illegible]rams

1. 1 mark for completed second row with test data of 11 characters:

Description	Test Type	Test Data
Test a value longer than the expected length.	Invalid	070000000000000
Test a value of the expected length.	**Normal**	**0700000000000**

2. 1 mark for each correctly completed row. For test one students can choose hig data must match the description. For test two students can choose a higher or match the description. For test three students can a value of a data type other

Description	Test Type	Test Data
Test the *highest/lowest* allowable integer	Boundary	*11/18*
Test a value *higher/lower* than allowable range	Invalid	*e.g. 5 or 22*
Test a value of an invalid data type	Erroneous	*e.g. "one"*
Test a value within the allowable range	Normal	*Any value between 11 and 18*

3. 11
4. `print(Message)`
5. 05 (also accept lines 06 and 08).
6. `if Gender == "Male" then`

   (also accept lines 06 [illegible] pped around).
7. 9 and 1[illegible]
8. `HumanAg[illegible]AgeConvert(CatAge)` (1)
   `print(Name + " is " + HumanAge + "in human years.")` (1)
9. 06
10. `return NewAge`

Boolean Logic

1. 1 mark per correct symbol

Gate	Symbol
NOT	
AND	
OR	

2. 1 mark for two different inputs, 1 mark for the correct outputs:

INPUT	OUTPUT
	1
	0

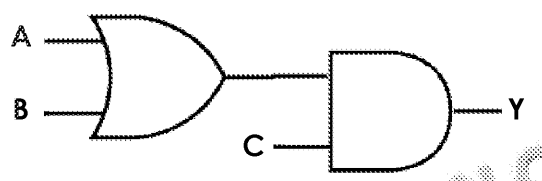
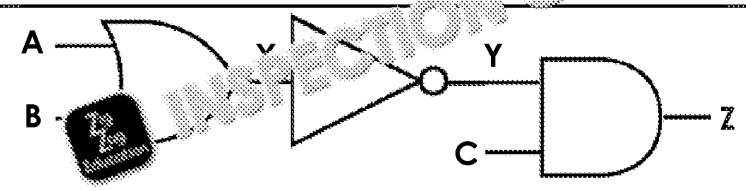
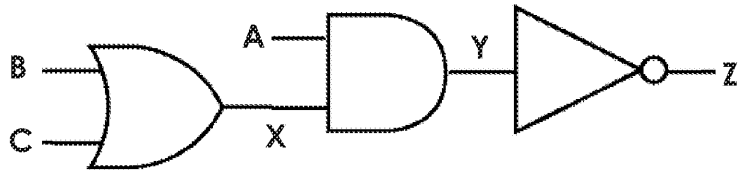
3. 1 mark 4 different combinations of inputs, 1 mark for each pair of correct outpu

INPUT 1	INPUT 2	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

4. 1 mark 4 different combinations of inputs, 1 mark for each pair of correct outpu

A	B	X	Y
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

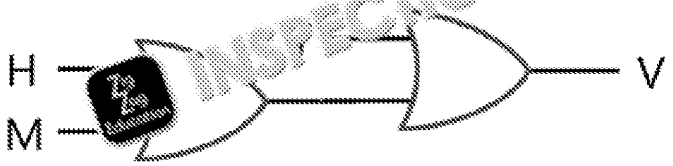
5. 1 mark for tick in correct row:

NOT(A AND (B OR C))	
	
	
	

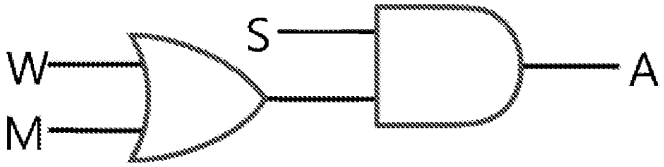
6. 1 mark 4 different combinations of inputs, 1 mark for each pair of correct outpu

A	B	C	X	
0	0	0	0	
0	0	1	1	
0	1	0	1	
0	1	1	1	
1	0	0	0	
1	0	1	1	
1	1	0	1	
1	1	1	1	

7. 1 mark for two of the inputs going into an OR ga 1 rk for the third input go for the output of the first OR gate going in he second OR gate.



8. 1 mark for inputs W and M going into an OR gate, 1 mark for the third input go the output of the OR gate going into the AND gate, 1 mark for correct logic sta



A = S AND (W OR M)

9. 1 mark 4 different combinations of inputs, 1 mark for each pair of correct outpu

W	M	S	W OR M	
0	0	0	0	
0	0	1	0	
0	1	0	1	
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	1	

Data Storage Units

It is important for yo
computer scienc

The smallest unit of storage is the bit. This can store a single 0 or 1.

Bits and Bytes

A lower-case b is used as a shorthand for bits, and an upper-case B is used as a shorthand for Bytes.

For example:

10 b = 10 bits

10 B = 10 Bytes

Co

You may be ask
for exa

For exam

10

We multiply by 1,000
result b

Converting from Bits

You may also be asked to give your answer
specific storag

For example, 8,000 bits into kilobytes:

8,000 / 8 / 1000 = 1 KB

We divide by 8 to convert to bytes and divide the result by 1,000 to get the answer in kilobytes.

Text St

We can calculate
usi

Bits per ch

For example, the sto
characte

Remember that spaces
inc

Image Storage Requirements

We can calculate the storage requirements of an image using the following formula:

Colour Depth × Width × Height

The resolution of an image is calculated by multiplying the width in pixels by the height in pixels.

For example, the storage requirements for an image with a width of 4, a height of 5 and a colour depth of 2 would be:

2 × 4 × 5 = 40 bits

Sound St

We can calculate t
usi

Bit Depths (bits)

For example, the stor
depth of 8 bits, a s

Number Representation

The denary number
symbols (0–9) to repres

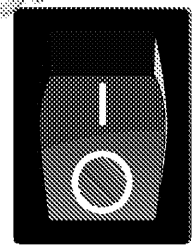
Humans us

Binary

Computers don't have fingers, they have circuits. These circuits can be in one of two states: on or off. So, they use a binary system.

On

Off



Cou

Each place
Thes

To convert a binary num
th

128	64	32	
0	0	0	

Counting in Binary

Each place in a binary number has a value.
These go up in multiples of 2.

To convert a binary number to denary we add up the place values of the columns with a 1 in.

128	64	32	16	8	4	2	1	
0	0	0	0	0	0	1	0	= 2

Counting in Binary

Each place in a binary number has a value.
These go up in multiples of 2.

To convert a binary number to denary we add up the place values of the columns with a 1 in.

128	64	32	16	8	4	2	1	
0	0	0	0	0	1	0	0	= 4

Counting in Binary

Each place in a binary number has a value.
These go up in multiples of 2.

To convert a binary number to denary we add up the place values of the columns with a 1 in.

128	64	32	16	8	4	2	1	
0	0	0	0	0	1	1	0	= 6

Counting in Binary

Each place in a binary number has a value.
These go up in multiples of 2.

To convert a binary number to denary we add up the place values of the columns with a 1 in.

128	64	32	16	8	4	2	1	
0	0	0	0	1	0	0	0	= 8

The leftmost bit in a binary number is known as the **most significant bit** as it has the highest value

The rightmost bit in a binary number is known as the **least significant bit** as it has the lowest value

De

Start by

10 =

128	64
0	0

Then write a 1 under

Finally, fill i

Hexadecimal

Hexadecimal is often used as a shorthand for binary as it is quicker for humans to write.

Hexadecimal is a base-16 number system, which means it uses 16 symbols.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

It starts by using the symbols 0--9 and then switches the letters A--F. This is so that only single digits are used.

F = 15

Hexa

Hexadecimal number
equ

Converting between hex
each character and co

Hex	
denary	
Binary	1 1

Finally, you joi

Hexadecimal to Binary

Hexadecimal numbers usually consist of pairs of digits as this is equivalent to 1 byte (8 bits).

Converting between hexadecimal and binary is simple. You just take each character and convert it to the equivalent binary number.

Hex	D	A
denary	13	10
Binary	1 1 0 1	1 0 1 0

Finally, you join the two binary numbers together.

DA = 11011010

Binar

To convert from binary to

Binary	1 1
denary	
Hex	

The easiest way to co
convert it to binary

Binary Arithmetic

Bi

The process of perf

Rules:
0 + 1 = 1
1 + 1 = 0 carry 1
1 + 1 + 1 = 1 carry

Rules Explained

Understanding the reasoning behind the rules can help you to understand how binary addition works.

Rules:
0 + 1 = 1
1 + 1 = 0 carry 1
1 + 1 + 1 = 1 carry 1

- 0 + 1 = 1 as we would expect.
- 1 + 1 = 2, which is 10 in binary.
- 1 + 1 + 1 = 3, which is 11 in binary.

When there isn't enoug
and it

No room for a carry,

Humans can easily work

Binary Shift

Binary shifts can be used to quickly multiply or di
numbers by powers of 2

A left shift of 1 wil ber by 2.

128	64	32	16	8	4	2	1	
0	0	0	0	1	0	0	0	= 8
0	0	0	1	0	0	0	0	= 16 (x2)
0	0	1	0	0	0	0	0	= 32 (x4)

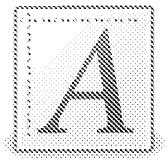
Any empty spaces are padded out with 0s.

Charact

Characters

Computers can only represent data using binary digits. In order to represent letters and symbols we assign each character a code.

Character codes are numbers which computers can represent using binary.



There are two main character sets used:

ASCII | Unicode

A comm
the **American Stand**

It uses seven bits for
character code. Seven
enough to code 128 di
characters.

128 character code
enough to represent
English alphabet plu
number of additional sy
such as punctuatio

Extended ASCII

Only supporting 128 characters makes ASCII very limited. Extended ASCII was developed to support additional characters.

It uses eight bits instead of seven to represent a character. This enables support for 256 characters.

Extended ASCII adds support for a number of additional characters including accented letters (e.g. à, é).

Character	Binary	Decimal
é	1000 0010	130
à	1000 0101	133
á	1010 0000	160
ó	1010 0010	162

Even extended ASCII
language in the world.

Unicode is an alterna
coding system that use
bits for each character

This enables 65536 diff
characters to be repres

Logical Ordering

The numbering of characters in character sets follo
order.

In ASCII lowercase a has
character c 7 a
subseque es
ch e.

There is a difference of 32 between upper and lowercase letter character codes. Subtracting 32 from a lowercase character code will give you the uppercase code.

Character	Binary	Decimal
a	0110 0001	97
b	0110 0010	98
c	0110 0011	99
d	0110 0100	100
e	0110 0101	101
f	0110 0110	102

B

Images can be repres
Each pixel has

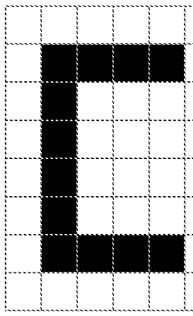
Colour Depth

A binary number is used to represent the colour of each pixel.

The number of bits used to store the colour of each pixel is known as the colour depth.

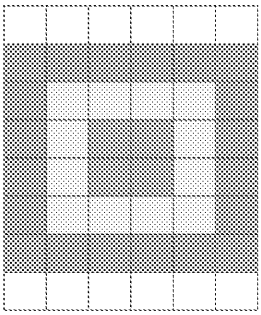
The greater the colour depth, the more colours can be represented.

An image with a colour
In this ex



2-Bit Images

An image with a colour depth of 2 bits can support four colours.
In this example 00 = White, 01 = Blue, 10 = Green, 11 = Yellow.



00, 00, 00, 00, 00, 00
10, 10, 10, 10, 10, 10
10, 11, 11, 11, 11, 10
10, 11, 01, 01, 11, 10
10, 11, 01, 01, 11, 10
10, 11, 11, 11, 11, 10
10, 10, 10, 10, 10, 10
00, 00, 00, 00, 00, 00

The resolution of an ima

The resolution of an i
calculated using th

Width (in pixels) * Hei

Therefore, the resolution

6 pixels * 8 pixels

Size and Quality

Increasing the colour depth and resolution can increa lity of an image but will also increase the stora re s

Increasing Colour Depth	Increasing Resolution
Enables more be and there mat the original s achieved.	Increases the amount of detail that is able to be represented.
It will also increase the storage requirements as each pixel requires more bits to store.	Increases storage requirements as there are more pixels to store.

Colour Depth: 4 bit
Resolution: 75 x 50
Size: 1.875 KB

Colour Depth: 24 bit
Resolution: 300 x 200
Size: 180 KB

Most digital images ha
the

Some metadata is ad
and some can a

An example of metada
height. Anoth

Many devices that capt
feature GPS chips whic
be tagged with the lo
taken

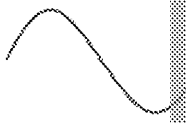
An example of metada
keywords th

Sound

A

Sound is analogue – t
wav

Computers can only wo

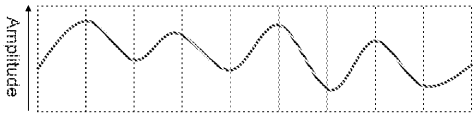


Original (analogue) sou

Sampling

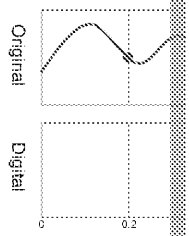
In order to digitise sound, computers have to take measurements of the **amplitude** (height of the wave) at regular intervals.

These measurements are known as **samples** and the process of digitising a sound is called **sampling**.



This is how many samp
it is also k

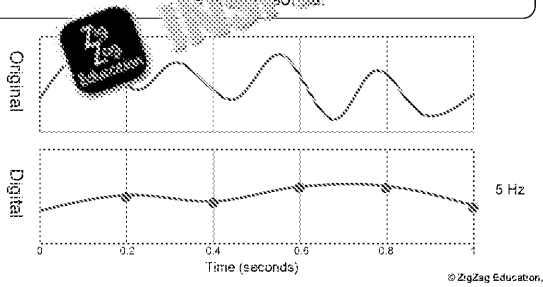
The higher the sample r



Sample Rate

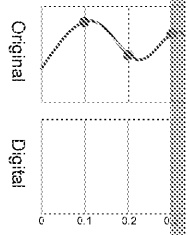
This is how many samples are taken per second (mea
it is also known as **sampling freq**

The higher the sample rate, th representation is to



This is how many samp
it is also k

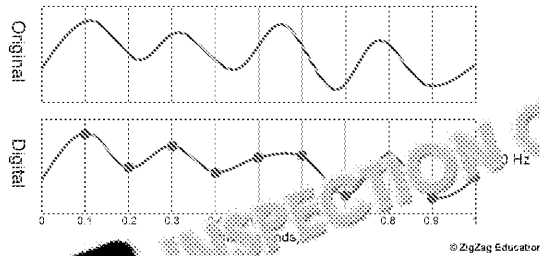
The higher the sample r



Sample Rate

This is how many samples are taken per second (measured in Hz) – it is also known as **sampling frequency**.

The higher the sample rate, the closer the digital representation is to the original sound.



Bit depth is the amoun

As with sample rate, t
represen

A bit depth of 8
(2^8 = 256) that c

A sample size of 16 bits
65536) that car

Bit Rate

The bit rate is the amount of storage required to store 1 second of sound. It is calculated using:

Sampling Frequency x Sample Size

For example a sound with a sample rate of 10 Hz and a bit depth of 16 bits would have a bit rate of 160 bits per second.

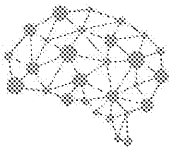
10 Hz x 16 bits = 160 bit/s

S

Increasing the samp
overall

It will also increase the

Computational Thinking



Comp

Computational thin
systematically an

The t

Abstraction

Abstraction

When analysing a problem it is necessary to **focus only on the important details** and discard what is not needed.
This technique is known as **abstraction**.

Here is an example problem with the important details highlighted:

There is a famous number guessing trick, which involves getting someone to think of a number and to perform some calculations on it in their head. From the result of the calculation anyone can work out what the original number was. The calculation involves doubling the number and adding 4, dividing by 2 and adding 7, multiplying by 8, subtracting 12, dividing by 4, taking away 11. The original number can be guessed by subtracting 4 from the result and dividing by 2.
Design a program that will get the user to think of a number, take them through the calculations, get the user to input the result and tells them the original number.

D

This is the process of t
more manage

This is an example of t

Ask
Guide u
Ask
Calcu
Outp

Algorithmic Thinking

This technique involves taking the parts of a decomposed problem and **working out the steps needed to achieve each part**.
These steps, when put together, form an algorithm.

Ask user to think of a number	`Ask user to think of number`

Guide user through the calculations	`Output Double the number and add 4`
	`Output Divide by 2 and add 7`
	`Output Multiply by 8, subtract 12`
	`Output Divide by 4, take away 11`

Ask user to input the result	`User to input result`
Calculate the original number	`Original number = (result - 4) / 2`
Output the original number	`Output original number`

Design

Analysing Problems

When analysing problems it can be helpful to first ... ts, processes and outputs in ...

Problem: a program is nee... of values, calculates the av... (m... ts the result to the user.

INPUT → PROCESS → OUTPUT

INPUT	PROCESS	OUTPUT
List of values (array)	Calculate average	Output average

Pr

Algorithms are ofte

Algorithms ca

Flow Charts

Flow Charts

Flow charts are used to represent algorithms visually, in the form of a diagram.

A number of standard symbols are used in flow charts:

Start/Stop — *Used to indicate the start or end of an algorithm.*

Process — *Used to indicate a process: for example, performing a calculation.*

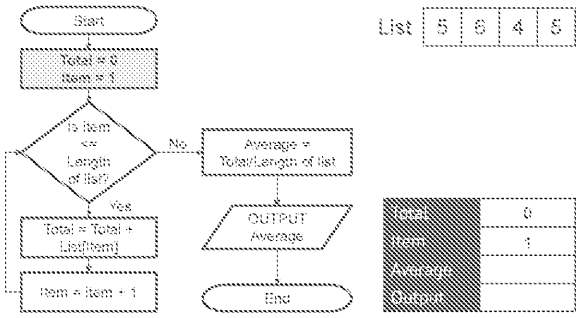
Input/Output — *Used when data needs to be inputted or outputted.*

Decision — *Used to control the path taken through an algorithm based on the result of a condition.*

Subroutine — *Used to call a defined subroutine.*

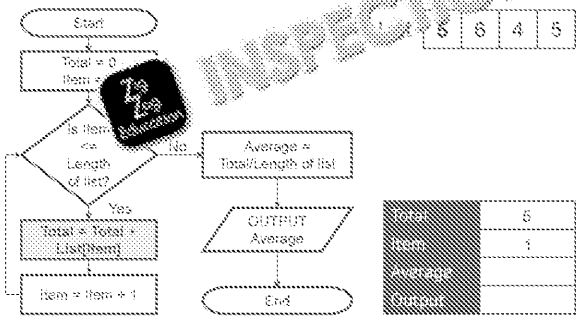
Example Flow Chart

This is an example algorithm represented using a flow chart, it is designed to calculate the average of a list of values.



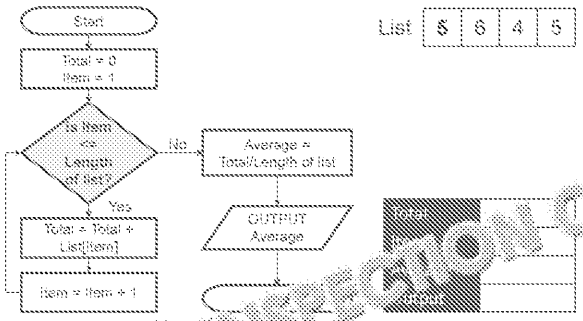
Example Flow Chart

This is an example algorithm represented using a flow chart, it is designed to calculate the average of a list of values.



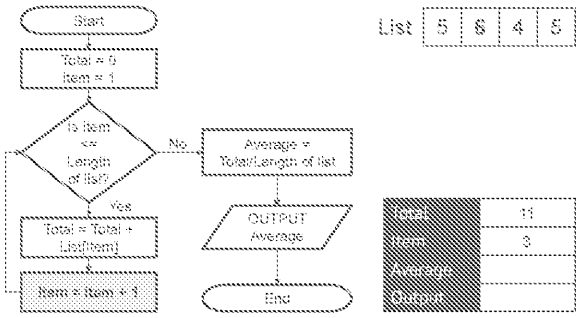
Example Flow Chart

This is an example algorithm represented using a flow chart, it is designed to calculate the average of a list of values.



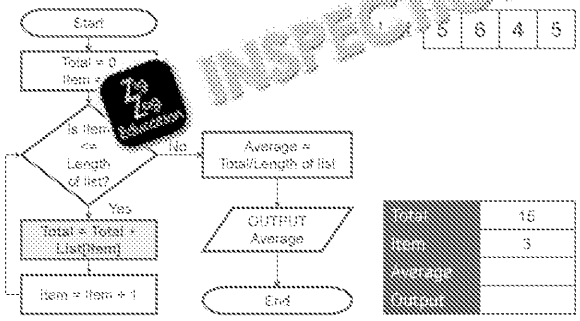
Example Flow Chart

This is an example algorithm represented using a flow chart, it is designed to calculate the average of a list of values.



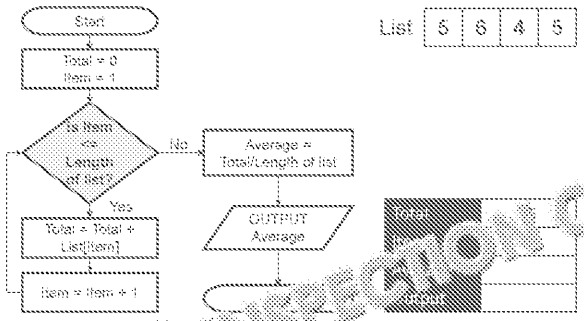
Example Flow Chart

This is an example algorithm represented using a flow chart, it is designed to calculate the average of a list of values.



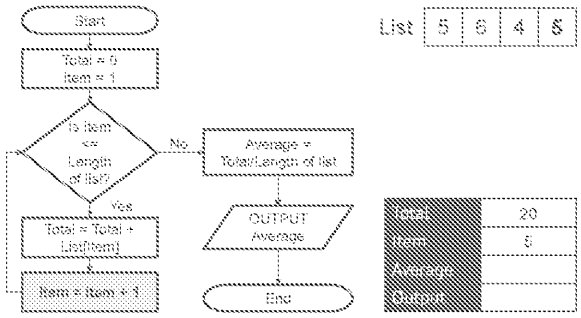
Example Flow Chart

This is an example algorithm represented using a flow chart, it is designed to calculate the average of a list of values.



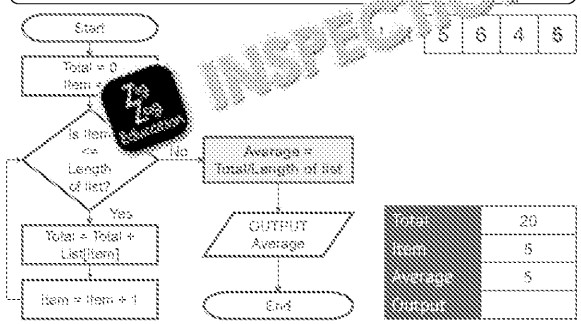
Example Flow Chart

This is an example algorithm represented using a flow chart, it is designed to calculate the average of a list of values.



Example Flow Chart

This is an example algorithm represented using a flow chart, it is designed to calculate the average of a list of values.



Pseudocode

Pseudocode is a way of designing an algorithm using a structure that looks vaguely similar to a real programming language.

There are no specific rules to writing pseudocode as it is informal; the important thing is to be consistent.

```
Set total to 0
Set item to 1
Get length of list
while item <= ListLength
      add current item to total
      increment item
End while loop
Divide total by length of list
Output calculated average
```

List	5	6	4	5

total	0
item	
ListLength	
average	
output	

Pseudocode

Pseudocode is a way of designing an algorithm using a structure that looks vaguely similar to a real programming language.

There are no specific rules to writing pseudocode as it is informal; the important thing is to be consistent.

```
Set total to 0
Set item to 1
Get length of list
while item <= ListLength
      add current item to total
      increment item
End while loop
Divide total by length of list
Output calculated average
```

List	5	6	4	5

total	0
item	1
ListLength	4
average	
output	

Pseudocode

Pseudocode is a way of designing an algorithm using a structure that looks vaguely similar to a real programming language.

There are no specific rules to writing pseudocode as it is informal; the important thing is to be consistent.

```
Set total to 0
Set item to 1
Get length of list
while item <= ListLength
      add current item to total
      increment item
End while loop
Divide total by length of list
Output calculated average
```

List	5	6	4	5

total	5
item	2
ListLength	4
average	
output	

Pseudocode

Pseudocode is a way of designing an algorithm using a structure that looks vaguely similar to a real programming language.

There are no specific rules to writing pseudocode as it is informal; the important thing is to be consistent.

```
Set total to 0
Set item to 1
Get length of list
while item <= ListLength
      add current item to total
      increment item
End while loop
Divide total by length of list
Output calculated average
```

List	5	6	4	5

total	11
item	1
listlength	
average	
output	

Pseudocode

Pseudocode is a way of designing an algorithm using a structure that looks vaguely similar to a real programming language.

There are no specific rules to writing pseudocode as it is informal; the important thing is to be consistent.

```
Set total to 0
Set item to 1
Get length of list
while item <= ListLength
      add current item to total
      increment item
End while loop
Divide total by length of list
Output calculated average
```

List	5	6	4	5

total	15
item	4
listlength	4
average	
output	

Pseudocode

Pseudocode is a way of designing an algorithm using a structure that looks vaguely similar to a real programming language.

There are no specific rules to writing pseudocode as it is informal; the important thing is to be consistent.

```
Set total to 0
Set item to 1
Get length of list
while item <= ListLength
      add current item to total
      increment item
End while loop
Divide total by length of list
Output calculated average
```

List	5	6	4	5

total	20
item	5
listlength	4
average	
output	

Pseudocode

Pseudocode is a way of designing an algorithm using a structure that looks vaguely similar to a real programming language.

There are no specific rules to writing pseudocode as it is informal; the important thing is to be consistent.

```
Set total to 0
Set item to 1
Get length of list
while item <= ListLength
      add current item to total
      increment item
End while loop
Divide total by length of list
Output calculated average
```

List | 5 | 6 | 4 | 5 |

P

Program code is the for
working progra

For example, here is th
average algorithm writt
in the Python
programming languag

Another Example

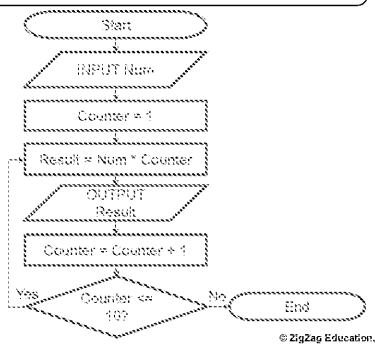
Now we are going to design an algorithm that will take a number and output the times table for it (from 1 to 10).

Ask the user to input a number, create a counter and set it to 1.

Multiply the number by the counter and output the result.

Add 1 to the counter.

Repeat while the counter is less than or equal to 10.



Exam

Here is example p

User
For

End

Structure Diagrams

Structure diagrams are a way of representing the
within a solution and the relationshi

Tracin

Tracing

Tracing allows us to manually test the inner workings of an algorithm or program to ensure everything works as intended.

Tracing is carried out using a trace table that has a column for each variable. There may also be a column for the output.

Each time a variable changes its value is placed in a new row of the trace table.

```
num = 5
for index = 1 to 3
    print(index*num)
next index
```

num	index	output
5		
	1	
		5
	2	
		10
	3	
		15

When tracing algorithm
expected to show the

If multiple variables ar
changed in one block
values can be placed in

```
array scores = [34,76
i = 0
count = 0
len = scores.length
while i < len
    if scores[i] >
        count =
    endif
    i = i + 1
endwhile
```

Sub Programs

Trace tables can also be used to test algorithms and program that feature sub programs.

```
function compare(num1, num2)
    largest = 0
    If num1 > num2 then
        largest = num1
    else
        largest = num2
    endif
    return largest
endfunction


result = compare(8, 12)
```

The values of the parameters along with the initial values of variables are placed in the first row.

num1	num2	largest	result
8	12	0	
		12	
			12

Sortin

Sorting Algorithms

A computer uses sorting algorithm nd lists of values.

T nu ifferent sorting algorithms; eed to know the following three:

Bubble Sort | Merge Sort | Insertion Sort

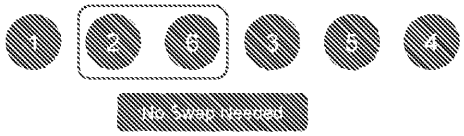
The bubble sort algorith
pairs and values and s

Pass 1: 2 1

Bubble Sort

The bubble sort algorithm works by working through a list, comparing pairs and values and swapping them if they are in the wrong order.

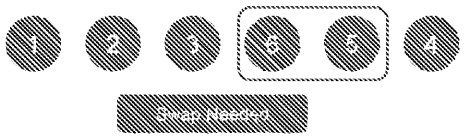
Pass 1



Bubble Sort

The bubble sort algorithm works by working through a list, comparing pairs and values and swapping them if they are in the wrong order.

Pass 1



Bubble Sort

The bubble sort algorithm works by working through a list, comparing pairs and values and swapping them if they are in the wrong order.

Pass 2

This process is repeated until there is a complete pass through the list without any swaps.

Bubble Sort

The bubble sort algorithm works by working through a list, comparing pairs and values and swapping them if they are in the wrong order.

Pass 2

No Swap Needed

This process is repeated until there is a complete pass through the list without any swaps.

Bubble Sort

The bubble sort algorithm works by working through a list, comparing pairs and values and swapping them if they are in the wrong order.

Pass 2

No Swap Needed

This process is repeated until there is a complete pass through the list without any swaps.

Bubble Sort Tracing

To show your understanding of the bubble sort algorithm, you may be asked to show the state of the list after each swap.

Original List	2	1	3	5	4	6
Swap 1	1	2	3	5	4	6
Swap 2	1	2	3	5	4	6
Swap 3	1	2	3	5	4	6
Swap 4	1	2	3	5	4	6
Swap 5	1	2	3	4	5	6

Alternatively, you could be asked to show the list after each complete pass.

Original List	2	1	3	5	4	6
Pass 1	1	2	3	5	4	6
Pass 2	1	2	3	4	5	6
Pass 3	1	2	3	4	5	6

Insertion Sort

The insertion sort algorithm uses two lists, one sorted and one unsorted.

Elements are gradually moved from the unsorted listed to the correct position in the sorted list.

Sorted

Unsorted

Insertion Sort

The insertion sort algorithm uses two lists, one sorted and one unsorted.

Elements are gradually moved from the unsorted listed to the correct position in the sorted list.

Sorted

Unsorted

Insertion Sort

The insertion sort algorithm uses two lists, one sorted and one unsorted.

Elements are gradually moved from the unsorted listed to the correct position in the sorted list.

Unsorted

Insertion Sort

The insertion sort algorithm uses two lists, one sorted and one unsorted.

Elements are gradually moved from the unsorted listed to the correct position in the sorted list.

Sorted | Unsorted

Inser

To show your understan
be asked to show h

Sorted			
			1
		1	2
	1	2	3
1	2	3	4

Insertion Sort Code

This is the insertion sort algorithm written in the OCR reference language.

```
for i=1 to list.length - 1
    pos = i
    while pos > 0 AND list[i] < list[pos-1]
        temp = list[i]
        list[pos] = list[pos+1]
        list[pos+1] = temp
        pos = pos - 1
    endwhile
next inext i
```

Goes through each element starting with the second.

Records a copy of the current element and its index.

Goes through the sorted portion of the list to find the correct place to insert the current element

You don't need to remember the code but you could be asked to identify the algorithm.

The merge sort algori
values and gradually m

7 2 1

Merge Sort

The merge sort algorithm works by splitting the list in
values and gradually merging them to form big
all in one sorted li

2 1 3 5 4 8

Next, merge the value and sorted pairs

The merge sort algori
values and gradually m

1 2 6

Next, merge t

Merge Sort

The merge sort algorithm works by splitting the list into individual values and gradually merging them to form bigger lists until they are all in one sorted list.

Finally, merge the groups of size 4 into a single list.

Mer

To show your understan
asked to show ho

Original List	7
Stage 1	7
Stage 2	2
Stage 3	1
Stage 4	1

Comparison

Each sorting algorithm has advantages and disadvantages.

Bubble Sort	This sorting algorithm is simple to implement, however, it is very inefficient.
Merge Sort	This sorting algorithm is one of the most efficient when used with both large and small lists
Insertion Sort	This sorting algorithm is relatively efficient when used with small lists, but not with larger lists

Searchi

Searching Algorithms

A computer uses searching algorithms to se a v in a given list.

Ther nu nt searching algorithms; ne know the following two:

Linear Search | Binary Search

L

The linear algo
the list one by o

In this exampl

2 1

The

The linear search algor
look

Linear Search Code

This is the linear search algorithm written in the OCR reference language.

```
function linearSearch(list, target)
    index = 0
    while index < list.length
        if list[i] == target then
            return i
        endif
        index = index + 1
    endwhile
    return -1
endfunction
```

Goes through each element in the list.

Compares the current element with the target and returns its index if it is found

Returns -1 if the target is not found

Y[illegible] to [illegible] number the [illegible] be asked to [illegible] the algorithm

B[illegible]

The binary search algo[illegible] however[illegible]

It starts by finding the m[illegible] searching fo[illegible]

1 2 [illegible]

Binary Search

The binary search algorithm is more efficient than the linear search; however, it only works on sorted lists.

It starts by finding the middle item and comparing it to the value it is searching for; in this case it is searching for 5.

1 2 3 4 5 6 7

If the middle value is smaller than the value it is searching for then the first half of the list is removed, including the middle value.

B[illegible]

The binary search algo[illegible] however[illegible]

It starts by finding the m[illegible] searching fo[illegible]

If the middle value is s[illegible] the first half of the li[illegible]

If the new middle valu[illegible] then the top half of the [illegible]

Binary Search

The binary search algorithm is more efficient than the [illegible] however, it only works on sorted [illegible]

It starts by finding the middle item [illegible] searching for: in [illegible] it [illegible] searching for 5.

5

If the middle value is smaller than the value it is searching for then the first half of the list is removed, including the middle value.

If the new middle value is larger than the value it is searching for then the top half of the list is removed, including the middle value.

This process is repeated until the value is found.

A[illegible]

If the list you want to se[illegible] won't be [illegible]

We find the midpoint of [illegible] dividing the length of [illegible] 8 ite[illegible]

We are now going t[illegible]

1 2 3

Another Example

If the list you want to search contains an even number of items, you won't be able to find an exact midpoint.

We find the midpoint of a list containing an even number of items by dividing the length of the list by two. In this example the list has 8 items, so the midpoint is 4.

We are now going to search for the value 3 in the list below.

1 2 3

Another Example

If the list you want to search contains an even number of items, you won't be able to find an exact midpoint.

We find the midpoint of a list containing an even number of items by dividing the length of the list by two. In this example the list has 8 items, so the midpoint is 4.

We are now going to search for the value 3 in the list below.

3

© ZigZag Education, 2020

Binary Search Code

This is the binary search algorithm written in the OCR refer[illegible]ge.

```
function binarySearch(list, ta[illegible]
    first = 0
    last = [illegible]list [illegible]
    [illegible]t
    [illegible] = (first + last) DIV 2
    [illegible] target == list[mid] then
        return mid
    else if list[mid] < target then
        last = mid - 1
    else
        first = mid + 1
    endif
endwhile
return -1
endfunction
```

Sets the pointers to first and last elements in the list.

Continues the search until pointers meet.

Finds the midpoint.

Compares the midpoint with the target and returns the position if they match.

Checks whether it is in the first half of the list and if it is the last pointer is moved to exclude the second half of the list.

Otherwise it must be in the second half so the first pointer is moved.

If the target is not found, -1 is returned.

You don't need to remember the code but you could be asked to identify the algorithm.

Program

Overview

Programs consist of a series of statements; a statement is a single instruction.

Programming constructs are used to control how the statements in a program are executed. There are three constructs:

- Sequence
- Selection
- Iteration

There are also a number of other concepts that are key to programming:

- Variables and Constants
- Inputs and Outputs
- Mathematical and Boolean Operators

This is the simplest pro
of statements that ar

This is an example pr
takes three number

```
Num1 = input("Enter
Num2 = input("Enter
Num3 = input("Enter
Total = Num1+Num2+Nu
Average = Total/3
print(Average)
```

Variables and Constants

A variable is a named memory location that can store a value which can be accessed and changed.

Constants also allow us to store values, however they cannot be changed while the program is running. This stops fixed values being changed accidentally.

The = operator is used to assign a value to a variable. This is known as assignment.

```
const ratio = 1.609
miles = input("How many miles?")
km = miles * ratio
print(km)
```

ratio is a constant that holds the ratio of miles to kilometres.

miles is a variable that stores the user's input.

km is a variable that stores the result of the conversion.

Inpu

In programming we of
and ou

There are examples

```
const ratio = 1.609
miles = input("How m
km = miles * ratio
print(km)
```

Selection

When this construct is used, the path taken through m
changes depending on the answer to a questio ate are
an example of select

This is an example pr ta wo numbers and either
e puts "They are equal".

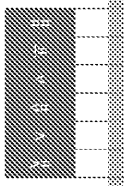
```
num1 = in er the 1st number")
num2 = inp nter the 2nd number")
if num1 == num2 then
      print("They are equal")
elseif num1 > num2 then
      print(num1)
else
      print(num2)
endif
```

Num1	10
Num2	5

Output	10

Comp

A number of different



Boolean Operators

Boolean operators are often used when writing conditions for selection statements.

AND	When used with two conditions, both must be met.
OR	When used with two conditions, one or both must be met.
NOT	Reverses the outcome of a condition.

This example program uses the OR Boolean operator to test whether the inputted age is within a given range.

```
age = input("Enter your age")
if age < 16 OR age > 65 then
      print("You are eligible for a discoun
else
      print("You are not eligib
endif
```


Arith

You will be familiar wi
programming and t

+	
-	S
*	M
/	
MOD	Modulus (divisio
DIV	Quotient (divi num
^	Exponent

Iteration

This construct is used to repeat a group of statements, avoiding the need to manually type out statements multiple times.

There are two types of iteration:

- Count-controlled
- Condition-controlled

Count

This construct is used t

This example prog
"H

```
for i=1 to 5
      print("Hello
next i
```

Count Controlled Loops

This construct is used to repeat a group of statements be of times.

This example program us output the text "H ve times.

```
for i=1 t
      print("Hello World")
next i
```

Index	1
Output	

Count

This construct is used t

This example prog
"H

```
for i=1 to 5
      print("Hello
next i
```

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
for i=1 to 5
      print("Hello World")
next i
```

Index	2
Output	Hello World Hello World

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
for i=1 to 5
      print("Hello World")
next i
```

Index	4
Output	Hello World Hello World Hello World Hello World

Count Controlled Loops

This construct is used to repeat a group of statements a set number of times.

This example program uses a **FOR loop** to output the text "Hello World" five times.

```
for i=1 to 5
      print("Hello World")
next i
```

Index	5
Output	Hello World Hello World Hello World Hello World Hello World

Condition Controlled Loops

Iteration can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
counter = 1
while counter <= 5
      print("Hello World")
      counter = counter + 1
endwhile
```

Counter: 1

Output:
Hello World

Condition Controlled Loops

Iteration can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
counter = 1
while counter <= 5
      print("Hello World")
      counter = counter + 1
endwhile
```

Counter: 3

Output:
Hello World
Hello World
Hello World

Condition Controlled Loops

Iteration can also be used to repeat a group of statements until a condition is met. A **WHILE loop** is an example of a condition controlled loop.

This example program uses a WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
counter = 1
while counter <= 5
      print("Hello World")
      counter = counter + 1
endwhile
```

Counter: 5

Output:
Hello World
Hello World
Hello World
Hello World
Hello World

Another Example

A **DO WHILE loop** is another example of a condition controlled loop. In this case the condition is placed at the end of the loop.

This example program uses a DO WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
counter = 1
do
    print("Hello World")
    counter = counter + 1
until counter > 5
```

Counter: 1

Output:

Another Example

A **DO WHILE loop** is another example of a condition controlled loop. In this case the condition is placed at the end of the loop.

This example program uses a DO WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
counter = 1
do
    print("Hello World")
    counter = counter + 1
until counter > 5
```

Counter: 2

Output:

Hello World

Another Example

A **DO WHILE loop** is another example of a condition controlled loop. In this case the condition is placed at the end of the loop.

This example program uses a DO WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
counter = 1
do
    print("Hello World")
    counter = counter + 1
until counter > 5
```

Counter: 3

Output:

Hello World
Hello World

Another Example

A **DO WHILE loop** is another example of a condition controlled loop. In this case the condition is placed at the end of the loop.

This example program uses a DO WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
counter = 1
do
    print("Hello World")
    counter = counter + 1
until counter > 5
```

Counter: 4

Output:
Hello World
Hello World
Hello World

Another Example

A **DO WHILE loop** is another example of a condition controlled loop. In this case the condition is placed at the end of the loop.

This example program uses a DO WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
counter = 1
do
    print("Hello World")
    counter = counter + 1
until counter > 5
```

Counter: 5

Output:
Hello World
Hello World
Hello World
Hello World

Another Example

A **DO WHILE loop** is another example of a condition controlled loop. In this case the condition is placed at the end of the loop.

This example program uses a DO WHILE loop to repeat the text "Hello World" until the Counter variable is greater than 5.

```
counter = 1
do
    print("Hello World")
    counter = counter + 1
until counter > 5
```

Counter: 6

Output:
Hello World
Hello World
Hello World
Hello World
Hello World

Data Types

When you create a variable, many programming languages require you to state the type of data you want to store in it.

Different types of data require different amounts of memory and, therefore, the computer needs to know how much memory to allocate to a particular variable.

The main data types are:

Integer | Real | Boolean | Char

Real numbers (also kn

Here are s

Temperature = 18.5

Boolean

The Boolean data type supports only two different values: True and False.

These values are represented using binary.

0 = False 1 = True

Overdue = False *Authorised = True*

The character data ty

This could

Here are

Gender = F

String

A string is a group of char

Strings are usu w quote marks " ".

are some examples of strings:

Name = "Alex" *Postcode = "N1 5BH"* *City = "London"*

Sometimes it is necessa
Th

For example, a numbe
converted to an int

The syntax for casting

Convert to string
Convert to integer
Convert to real
Convert to Boolean

Casting Example

This example program is designed to convert inches to centimetres and output the result.

It casts the resulting calculation to a string so it can be joined with another string in the output.

```
const ratio = 2.54
inches = input("How many inches?")
cm = inches * ratio
strCm = str(cm)
print(strCm + " cm")
```

The result of this calculation will produce a **real** value.

This casts the **real** value to a **string**.

String

String Manipulation

Most programming languages feature a range of in-built operations that allow us to manipulate strings.

Common operations include:

- Concatenation
- Substring
- Length
- Case Conversion
- Character/Code Conversion

C

Concatenation is the pr
The + ope

This example program a
their name to the string
the **message** variable.

```
name = input(
message = nam
print(message
```

If a string and an intege
form a string. This is kn
used in all programmin

Length

Most programming languages have a way of calc ng th
of a string. In OCR reference language we method.

This example program asks to t some text. It then finds
the length stri t in the **textLength** variable.
F put the length of the text to the screen.

```
text = input("Enter some text")
textLength = text.length
print(textLength)
```


The substring methods c

Index	0
Character	H

Number of char

```
message.substring(1
```

Substring position

Number of c

```
message.left(2)
```

This example would extract "
the string stored in the message

Case Conversion

Many programming languages feature methods that allow us to convert strings to upper or lower case.

The following would output "HELLO WORLD" followed by "hello world".

```
message = "Hello World"
print(message.upper)
print(message.lower)
```


Charact

Each character is rep
there are methods th

ASCI("a")

This method will conv
character to its ASCII c
this example 97 woul
outputted.

Data Structures and File Handling

D

When programming, it

One option is to use mu
could be used. A data
unc

The

Arrays

An array allows a set of related values to be stor a
list under one identif

Each position in an arr ex which allows it to be
ed.

This is at stores the top scores for an online game.

Index	0	1	2	3	4
Value	24	98	21	44	76

This value could be accessed using: scores[2]

An array allows a set o
li

Each position in an ar

This is an array that

Index	0
Value	24

Creating an Array

In OCR reference language we create an array by placing the values in square brackets separated by commas.

The code for creating the scores array is shown below:

Index	0	1	2	3	4
Value	24	98	21	44	76

```
array scores = [24, 98, 21, 44, 76]
```

An empty array of a set size can also b[illegible]ted

```
array
```


Two-d[illegible]

It is possible to represe[illegible]

An example of a [illegible]

Index	0
0	24
1	87
2	65

Two-dimensional Arrays

It is possible to represent a table of values using a two-dimensional (2D) array.

An example of a two-dimensional array is shown below:

Index	0	1	2	3	4
0	24	98	21	44	76
1	87	29	7	21	88
2	65	45	31	77	35

This value can be accessed using scores[2, 1]

This value can be changed using scores[1, 4] = 37

Cre[illegible]

In code we create a 2D [illegible]

The code for c[illegible]

Index	0
0	24
1	87
2	65

Populated array: arra[illegible] [24, [illegible]

Empty array: arra[illegible]

Arrays and FOR Loops

A FOR loop can be used to cycle through each valu[illegible]y.

This example can be used to cycle [illegible]gh e[illegible]alue in the 1D [illegible]

Index	0	1	2	3	4
Value	[illegible]	98	21	44	76

score: 24

Output:

```
for i=0 to scores.length-1
    print(scores[i])
next i
```


Arra[illegible]

A FOR loop can be us[illegible]

This example can b[illegible]

Index	0
Value	24

score: 24

Output:

24

Arrays and FOR Loops

A FOR loop can be used to cycle through each value in an array.

This example can be used to cycle through each value in the 1D scores array.

Index	0	1	2	3	4
Value	24	98	21	44	76

Score: 98

Output:
24

```
for i=0 to scores.length-1
    print(scores[i])
next i
```


Arrays and FOR Loops

A FOR loop can be used to cycle through each value in an array.

This example can be used to cycle through each value in the 1D scores array.

Index	0	1	2	3	4
Value	24	98	21	44	76

Score: 21

Output:
24
98

```
for i=0 to scores.length-1
    print(scores[i])
next i
```


Arrays and FOR Loops

A FOR loop can be used to cycle through each value in an array.

This example can be used to cycle through each value in the 1D scores array.

Index	0	1	2	3	4
Value	24	98	21	44	76

Score: 44

Output:
24
98
21

```
for i=0 to scores.length-1
    print(scores[i])
next i
```

Arrays and FOR Loops

A FOR loop can be used to cycle through each value in an array.

This example can be used to cycle through each value in the 1D scores array.

Index	0	1	2	3	4
Value	24	98	21	44	76

Score
76

Output
24 98 21 44

```
for i=0 to scores.length-1
    print(scores[i])
next i
```


Arrays

A FOR loop can be us

This example can b

Index	0
Value	24

Score
76

Output
24 98 21 44 76

Databases

A database is a collection of data that is grouped together in an organised way. Databases store data in tables, this is an example of a table that is used to store student details.

StudentID	FirstName	LastName	Address	DOB	Gender
1	John	Curtis	12 Brook Lane	21/03/1990	Male
2	Ben	Jackson	1 Totters Lane	15/04/1990	Male
3	Sarah	Smith	60 Belsize Rd	6/06/1990	Female

Each individual piece of information in a table is known as a field, for example LastName is a field.

A database is a colle
organised way. Databas
a table that

StudentID	FirstName	L
1	John	C
2	Ben	Ja
3	Sarah	S

Each individual piece of
exa

All the information rel
record, for exam

Records

Usually specialised software is used to create and
databases. However, a 2D array can be used
with each 1D array forming an individual

Here is an example of
stored using a 2D array:

1	Bob	15001	03/05/16
2	Carol	21023	14/06/16
3	Steve	18730	19/06/16

```
array members = [["1","Bob","15001","03/05/16"],
["2","Carol","21023","14/06/16"],
["3","Steve","18730","19/06/16"]]
```


Sometimes it is neces
be kept after

Here is an example

```
members.txt
1, Bob, 15001, 03/0
2, Carol, 21023, 14
3, Steve, 18730, 19
```

```
file = open("member
while NOT file.end
    print(file.re
endwhile
file.close()
```

Creating a File

This example program shows how a new file can be created and data written to it.

```
newFile("scores.txt")
file = open("scores.txt")
array scores = [5, 7, 2, 4, 9]
for i = 0 to scores.length-1
      file.writeLine(scores[i])
next i
file.close()
```

```
scores.txt
```


Creating a File

This example program shows how a new file can be created and data written to it.

```
newFile("scores.txt")
file = open("scores.txt")
array scores = [5, 7, 2, 4, 9]
for i = 0 to scores.length-1
      file.writeLine(scores[i])
next i
file.close()
```

```
scores.txt
5
7
```

1

Creating a File

This example program shows how a new file can be created and data written to it.

```
newFile("scores.txt")
file = open("scores.txt")
array scores = [5, 7, 2, 4, 9]
for i = 0 to scores.length-1
      file.writeLine(scores[i])
next i
file.close()
```

```
scores.txt
5
7
2
4
```

3

Sub Programs

A sub program is a sect
can be called whe

Sub programs are use
programs and

There are tw

Procedures

Procedures

A procedure is a sub program that doesn't return a value back to the main program.

This is an example procedure that will output the text "Computer", "Science", "Rocks" whenever it is called.

A procedure or function won't run unless it is called.

```
procedure message()
    print("Computer")
    print("Science")
    print("Rocks")
endprocedure

message()
```

Procedure call

Output

```
Computer
Science
Rocks
```


Sometimes procedures
their job. Th

Parameters are like p
function expe

In this example the val
Height parameters

```
procedure areaCalc(w
    area = width * he
    print(area)
endprocedure

areaCalc(10, 5)
```

Parameters

Sometimes procedures and functions need data in order to
their job. This is where parameters come in.

Parameters are like placeholde
procedure or
function expects
when it is called.

In this ex
val
10 and 5 are passed into the Width and
Heigh
rs when the areaCalc procedure is called.

Parameters

```
procedure areaCalc(width, height)
    area = width * height
    print(area)
endprocedure

areaCalc(10, 5)
```

width	10
height	5
area	50

Output

```
50
```


A function is a sub prog

The returned value take

This is the areaC

```
function areaCalc(w
    area = width * he
    return area
endfunction

print(50)
```

Variable Scope

Where a variable can be accessed within a program depends on the variable scope. There are two types of variable:

Global	Declared outside any sub programs and accessible throughout the program, including inside sub programs
Local	Declared within a sub program and only accessible within that sub program.

This example program features a procedure to calculate speed.

It uses both a local and a global variable.

```
procedure speedCalc(distance, time)
        speed = distance / time
        return speed
endprocedure

answer = speedCalc(distance, 90)
```


(Structur

Introduction

Structured Query Language (SQL) is used to search databases using queries.

Queries are used to manipulate the data in a database; for example, searching, adding, editing or deleting records.

```
SELECT First_Name, DOB
FROM Student
WHERE Gender = "Male"
```

*The **SELECT** command is used to choose which fields you want to show.*

Student ID	First Name	Last Name	Address	DOB	Gender
1	John	Curtis	12 Brook Lane	21/03/1990	Male
2	Ben	Jackson	1 Totters Lane	15/04/1990	Male
3	Sarah	Smith	60 Belsize Rd	06/06/1990	Female

Structured Query Lan

Queries are used to ma
searching, a

```
SELECT First_Name,
FROM Student
WHERE Gender = "Male
```

Student ID	First Name
1	John
2	Ben
3	Sarah

Introduction

Structured Query Language (SQL) is used to search
using queries.

Queries are used to manipulate
searching, adding
editing records.

```
SELECT F
FROM Stud
WHERE Genc
```

*The **WHERE** command is used to filter the results.*

Student ID	First Name	Last Name	Address	DOB	Gender
1	John	Curtis	12 Brook Lane	21/03/1990	Male
2	Ben	Jackson	1 Totters Lane	15/04/1990	Male
3	Sarah	Smith	60 Belsize Rd	06/06/1990	Female

Structured Query Lan

Queries are used to ma
searching, a

```
SELECT First_Name,
FROM Student
WHERE Gender = "Male
```

Student ID	First Name
1	John
2	Ben
3	Sarah

Introduction

Structured Query Language (SQL) is used to search databases using queries.

Queries are used to manipulate the data in a database; for example, searching, adding, editing or deleting records.

```
SELECT First_Name, DOB
FROM Student
WHERE Gender = "Male"
```

Here are the results of this query:

First_Name	DOB
John	21/03/1990
Ben	15/04/199

ORDER BY is used t

```
SELECT *
FROM Student
WHERE Gender = "Male
ORDER BY Last_Name D
```

Student ID	First_Name	
1	John	
2	Ben	
3	Sarah	

ORDER BY

ORDER BY is used to set the order in which the results appear.

```
SELECT *
FROM Student
WHERE Gender = "Male"
ORDER BY Last_Name DESC
```

The results of this query will be displayed in descending order by the Last_Name field.

Student ID	First_Name	Last_Name	Address	DOB	Gender
2	Ben	Jackson	1 Totters Lane	15/04/1990	Male
1	John	Curtis	12 Brook Lane	21/03/1990	Male

```
SELECT *
FROM Student
WHERE Gender = "Male"
ORDER BY Last_Name
```

Excluding the DESC command will result in the records being displayed in ascending order.

Coml

Multiple conditions c

```
SELECT *
FROM Assessment
WHERE Test_1 >= 50 A
```

This query will return a

Student ID	Test_1
3	100

LIKE

The **LIKE** operator can be used with the **WHERE** c nd results that meet a certain

```
SELECT *
FROM Stu
WHERE Fi LI A
```

e character is used as a wildcard, this can be used to represent any sequence of characters.

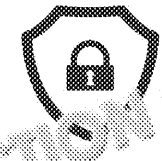
Student ID	First_Name	Last_Name	Address	DOB	Gender
2	Ben	Jackson	1 Totters Lane	15/04/1990	Male
3	Sarah	Smith	60 Belsize Rd	6/06/1990	Female
4	Susan	Wright	8 Canonbury Rd	18/09/1990	Female

The **LIKE** operator car results

```
SELECT *
FROM Student
WHERE First_Name LIK
```

Student ID	First_Name	
3	Sarah	
4	Susan	

Defensive Design



De

When designing prog
robust by anticipatin
maintainable to ens

There are thr

Input Validation

Input Validation

It is important to anticipate that users of your program, intentionally or mistakenly, are likely to input data that it is not designed to accept.

This is where input validation comes in. It enables programs to check that data entered meets certain rules.

Types of validation include:

Range Checks | Length Checks

R

Range checks are used

This example code che

```
mark = input()
if mark >= 0 AND mar
      print("Mark v
else
      print("Mark o
endif
```

Length Checks

Length checks are used to test whether the value e a specified length.

This example code checks wh d entered is at least o length.

```
password Enter password: ")
if passwor ength < 6 then
      print("Password too short")
else
      print("Password valid")
endif
```

Input	"Hello"
Output	"Password too short"

A

For programs that handle personal or sensitive data it if often necessary to include user authentication. This is usually achieved by requiring a username and password.

This example code will not allow the user to continue until they enter a valid password.

Authentication

For programs that handle personal or sensitive data it if often necessary to include user authentication. This is usually achieved by requiring a username and password.

This example code will not allow the user to continue until they enter a valid password.

```
password = "n2t2"
valid = FALSE
while valid == FALSE
        usrPass = input("Password: ")
        if usrPass == password then
                valid = TRUE
                print("Logged In")
        else
                print("Invalid")
        endif
endwhile
```

Input	
password	"n2t2"
valid	
Output	

Authentication

For programs that handle personal or sensitive data it if often necessary to include user authentication. This is usually achieved by requiring a username and password.

This example code will not allow the user to continue until they enter a valid password.

Authentication

For programs that handle personal or sensitive data it if often necessary to include user authentication. This is usually achieved by requiring a username and password.

This example code will not allow the user to continue until they enter a valid password.

```
password = "n2t2"
valid = FALSE
while valid == FALSE
        usrPass = input("Password: ")
        if usrPass == password then
                valid = TRUE
                print("Logged In")
        else
                print("Invalid")
        endif
endwhile
```

Input	"n1t2"
password	"n2t2"
valid	FALSE
Output	

Authentication

For programs that handle personal or sensitive data it if often necessary to include user authentication. This is usually achieved by requiring a username and password.

This example code will not allow the user to continue until they enter a valid password.

Authentication

For programs that handle personal or sensitive data it if often necessary to include user authentication. This is usually achieved by requiring a username and password.

This example code will not allow the user to continue until they enter a valid password.

```
password = "n2t2"
valid = FALSE
while valid == FALSE
        usrPass = input("Password: ")
        if usrPass == password then
                valid = TRUE
                print("Logged In")
        else
                print("Invalid")
        endif
endwhile
```

Input	"n2t2"
password	"n2t2"
valid	FALSE
Output	"Invalid"

Authentication

For programs that handle personal or sensitive data it if often necessary to include user authentication. This is usually achieved by requiring a username and password.

This example code will not allow the user to continue until they enter a valid password.

Authentication

For programs that handle personal or sensitive data it if often necessary to include user authentication. This is usually achieved by requiring a username and password.

This example code will not allow the user to continue until they enter a valid password.

```
password = "n2t2"
valid = FALSE
while valid == FALSE
      usrPass = input("Password: ")
      if usrPass == password then
            valid = TRUE
            print("Logged In")
      else
            print("Invalid")
      endif
endwhile
```

input	"n2t2"
usrPass	"n2t2"
valid	TRUE
print	"Logged In"

M

It is important to ensu that

Programmers often wor to edit and

Programmers also oft date, using good progr becom

There are four main te ma

Hello Sub Program

Na Cons

Using Sub Programs

To make code easier to understand and maintain it should be broken down into sub programs.

Each sub program should perform a separate function, enabling them to be worked on individually.

Teams of programmers can work on a program together by each member being assigned a separate sub program to develop.

Nam

It is important to ma programs have meanin Single letter variable An exception to this

Choosing a naming c particularly

Spaces cannot be used alternative metho The

Underscores: total_score

Comments and Indentation

Comments are designed to help the programmer u e function of each section of code and are i outer.

The use of indentation helps ich blocks of code are each other.

```
//asks th input their mark
Mark = inp
//checks whether the mark is valid
if Mark >= 0 AND Mark <= 25 then
      print("Mark valid")
else
      print("Mark out of range")
endif
```

Use of commenting.

Use of indentation

Testi

Testing

It is important to test our programs to ensure they work as we expect them to. There are two main types of testing:

Iterative Testing	Carried out during development, usually on individual sub programs or modules.
Final/Terminal Testing	Carried out at the end of the development process. Tests the program as a whole.

When testing our pr

There are four types testing

Normal	Bo
Data that is well within the normal range of what should be accepted.	Data the type the b of sho acc

Test Table Example

```
mark = input()
if mark >= 0 AND mark <= 25 then
      print("Mark valid")
else
      print("Mark out of range")
endif
```

Description	Test Type	Test Data	Expected Outcome	Result
Test the highest allowable integer	Boundary	25	'Mark valid'	'Mark valid'
Test value higher than allowable range	Invalid	30	'Mark out of range'	'Mark out of range'
Test Value within allowable range	Normal	10	'Mark valid'	'Mark valid'

1

Errors or 'bugs' occu exp

Being able to identify

There are main t

Syntax

Syntax Errors

Syntax errors occur when part of the code **breaks th** the programming language

This could be as simple as for a closing bracket or forgetting to in otation marks.

```
      put("Enter the first number"
      put("Enter the second n
if num1 < num2 then
      print(They are equal)
elsif num1 = num2 then
      print(num1)
      print(num2)
endif
```

Missing bracket

Missing quotes

Mistyped command word

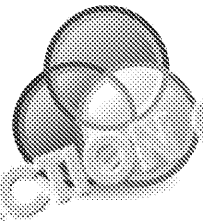
A **logic error** occurs w error messages b

This is due to an error simple as using the wr

```
num
num
if
els
els
end
```

Conversion operator in the wrong way round

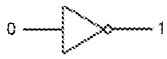
Boolean Logic



Logic gates are the fu

There

NOT

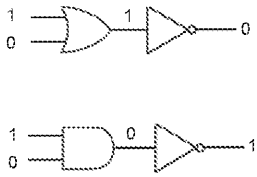


The NOT gate reverses the input, so if 1 is inputted into a NOT gate it will output 0.

Combining Gates

Logic gates can be combined in order to create logic circuits.

For example:



T

All the possible outco

We start by filling i

A
0
0
1
1

Truth Tables (2)

Here is another truth table for a logic diagram featurin gic gates.

When the ple gates another column is sometimes added to show intermediary results.

A	B	X	Y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Comb

When completing truth t all the po

Logic diagrams with two need truth tables with fou

Start with the second inp alternate 0s and 1s

For the first input we alt pairs of 0s and 1s

When there are three i there should be eight ro the first input alternating of four 0s and 1s

Logic Statements

Here is an example of a logic statement:

C = A OR B

This logic statement shows that C is 1 if either or both A and B are 1.

We can also combine Boolean operators to create more complex logic statements.

C = NOT(A OR B)

In this case the NOT operator is applied to the [illegible] of [illegible] this means it is reve[illegible]

You may be asked to p[illegible]

- A system is used to m[illegible]
- A heart rate (H) moni[illegible]
- An oxygen (O) is use[illegible]
- An alarm (A) is soun[illegible] readings are outside [illegible]