EDULiTO

Practice Exams 2

GCSE (9–1) Computer Science

Edexcel

1CP2/02 Application of Computational Thinking

Time allowed: 2 hours

Question paper

First name(s)	
Last name	
Class	

You must have:

- a computer workstation with a Python IDE that you are familiar with.
- a folder containing code and data files.

Instructions

• Answer all the questions on your computer.

• Save new or amended code using the file name provided and place it in the 'COMPLETED CODE' folder.

• You must not use the internet at any time during the examination.

Information

• The folder in your user area includes all the code and data files you need.

• The total mark for this paper is 75.

• The marks for each question are shown in brackets.

Advice

• Read each question carefully before you start to answer it.

• Save your work regularly.

Answer ALL questions.

Suggested time: 10 minutes

1. A program creates product codes for a business. The first design of the program is shown in the flowchart in Fig. 1.

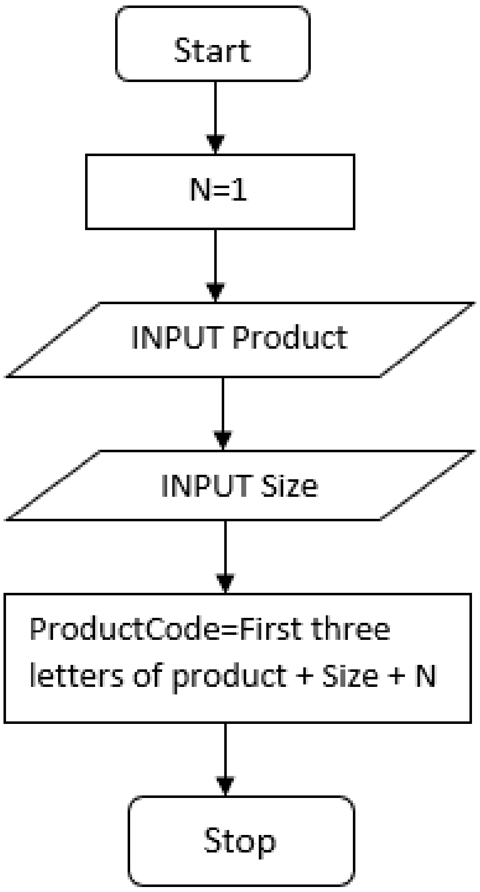


Figure 1

(a) Open file **Q01**. Develop the program shown in the flowchart using Python. The first product to be coded is “jumper” and is size S (small).

Amend the code to add or complete lines to:

- include **four** variables (product, size, n, product_code)
- create an input statement so that the user can enter a product name
- create an input statement so that the user can enter the size of the product
- cast the data type for n from integer to string
- create a product code using the criteria in the flowchart
- output the product code.

Do not add any additional functionality. Save your amended code file as **Q01FINISHED.py**

(Total for Question 1 = 7 marks)

Suggested time: 10 minutes

2. A programmer has started to write a program, but it does not work correctly. It is a fortune teller game that looks into your future and provides you with a prediction.
Open file **Q02**.

Amend the code to:

- fix the syntax error on line 4
- fix the syntax error on line 6
- fix the syntax error on line 10
- change the identifier **y** to a more meaningful name
- add a seventh prediction to the game
- add a comment to explain how the random module is used.
- add two uses of white space to aid readability.

Do not add any additional functionality. Save your amended code file as **Q02FINISHED.py**

(Total for Question 2 = 7 marks)

Suggested time: 15 minutes

3. Open file **Q03a**. This Python program has been written to add numbers together. It will add as many numbers as required by the user.

The program:

- continues until reply is equal to “N”
- inputs a number entered by the user
- adds the number to the total
- asks the user whether they would like to add another number
- if the user decides to enter no more numbers then the user is presented with their total.

Amend the code to:

- include indentation
- improve the program’s validation, so that the program accepts an upper case “Y” or a lowercase “y” response
- fix the logic errors that causes the total to be equal to the last number that has been entered
- fix the logic error that causes the program to output the last number rather than the total.

Do not add any additional functionality. Save your amended code file as **Q03aFINISHED.py**

(b) Open file **Q03b.** The programmer has decided to use a count-controlled loop so that exactly 3 numbers can be added together each time the program is run.

Write the updated program using file **Q03b**.

Do not add any additional functionality. Save your amended code file as **Q03bFINISHED.py**

(Total for Question 3 = 9 marks)

Suggested time: 20 minutes

4. A vet has decided to produce a Python program that will provide dog owners with a human age equivalent for their dog's age in years. For example, using this program a dog that is 5 years old will be shown to have a human equivalent age of 36.

The program:

- uses a suitable identify for a variable that stores a dog's age.
- inputs from the user the age of the owner's dog using a suitable data type.
- uses a suitable identify for a variable that stores human age equivalent.
- calculates that if the age of the dog equals 1 then the human age equivalent is 12.
- calculates that if the age of the dog is greater than 1 then the human age equivalent

  is equal to 24 + (4 x dog age - 2)
- outputs the human equivalent age for the owner.

(a) Open file **Q04a** and develop this Python program using the information you have been given.

Do not add any additional functionality. Save your amended code file as **Q04aFINISHED.py**

(b) The vet decides to adapt the program so that it uses a function to carry out the calculation and returns the human equivalent age to the main program. The function is called **dog_years** and it uses the parameter **age**.

Update the program so that:

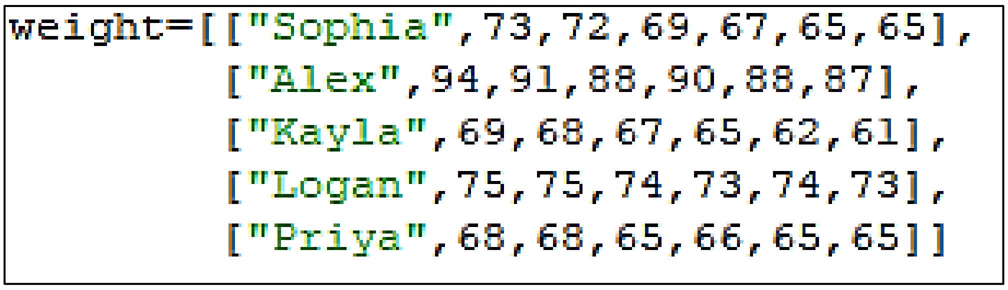
- the function is defined
- the body of the function to perform the calculations.
- the human equivalent age is returned to the main program
- the main program asks the owner to enter the age of the dog
- the main program calls the function
- the main program outputs the human equivalent age of the dog.

Do not add any additional functionality. Save your amended code file as **Q04bFINISHED.py**

(Total for Question 4 = 13 marks)

Suggested time: 20 minutes

5. A dietician has a number of patients aged under 18. She has designed a program to monitor the weight (in Kilograms) of these individuals over a six-month period. Data for each month (from January to June) is stored in a 2D array (list) with the identifier `weight`.



```
weight=[["Sophia",73,72,69,67,65,65],
        ["Alex",94,91,88,90,88,87],
        ["Kayla",69,68,67,65,62,61],
        ["Logan",75,75,74,73,74,73],
        ["Priya",68,68,65,66,65,65]]
```

We can see from the 2D array that Sophia weighed 73 kg in January, 67 kg in April and 65 kg in June.

Therefore **`print(weight[0][2])`** produces the output **72.**

(a) Open the file **Q05a**. The dietician's program outputs the name of each person and their weight at the end of the sixth month period, using the 2D array (list).

Refine the program to be **more efficient**. The program must produce the same output using two lines of code (excluding the list)

Do not add any additional functionality. Save your amended code file as **Q05aFINISHED.py**

(b) The dietician decides to improve her program further to show the total weight loss of each person at the end of the six-month period.

Open the file **Q05b**. Develop a program that:

- is efficient
- calculates the total weight loss of each person i.e. their weight in January minus their weight in June.
- Only outputs the name and total weight loss at the end of the six-month period of individuals who lost more than 4 kg.

Do not add any additional functionality. Save your amended code file as **Q05bFINISHED.py**

(Total for Question 5 = 10 marks)

Suggested time: 20 minutes

6. You have been asked to develop a Python program that manages football scores.

(a) Open the file Q06a. Develop a program that:

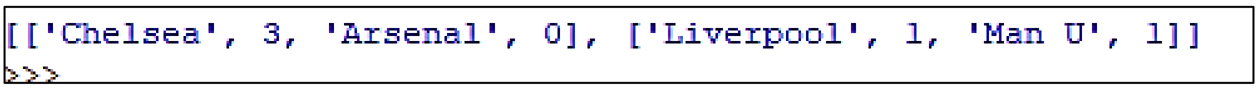
- inputs the name of each team that has just played a match.
- input the score achieved by each team in this particular game
- use a comparison to determine which team has won the game
- output the name of the team who has won the game or in the case of a draw, output "draw".

Do not add any additional functionality. Save your amended code file as **Q06aFINISHED.py**

(b) Open file **Q06b**. You have been asked to develop the program further to store the name of each team and the points achieved in each game (3 points for a win, 1 point for a draw and 0 points for a loss) to a 2D list.

The program:

- includes the functionality of **Q06a**.
- includes a 1D list to store the name of each team and the points relating to a particular game. This list has the identifier "new_entry"
- includes a 2D list to store the results from all matches that have been entered. This list has the identifier "results"
- has a variable with the identifier "entry" to store the value "Y" or "N"
- has a condition-controlled loop, based on whether the user would like to enter the results for another match. The user can make as many entries as they would like.
- appends the name of each football team and the points each team has achieved to the new_entry list.
- appends the new_entry list to the results list.
- Outputs the results list. Here is an example of a results list that stores two matches:



```
[['Chelsea', 3, 'Arsenal', 0], ['Liverpool', 1, 'Man U', 1]]
>>>
```

Do not add any additional functionality. Save your amended code file as **Q06bFINISHED.py**

(Total for Question 6 = 14 marks)

Suggested time: 25 minutes

7. You have been asked to develop a Python program dice game. Before the player can start the game, they must be authenticated by entering their name which is compared with the names stored in a list. Once authenticated they can play the game. In the game two dice are thrown. If the player gets a total of more than 8, they get another throw of the two dice. If they get a double six, they have won.

Open file **Q07**.

Write a program to meet the following requirements:

Inputs

- Prompt for a name which can be entered by the user.

Process

- The program accepts lowercase entries of a name and capitalizes the entry to match the list.
- The list `players` is searched to see whether the name entered matches a name in the list.
- If the name is not authenticated the player is asked to enter a name. This is repeated until an authenticated name is entered.
- If the player's name is authenticated the game starts.
- Each dice randomly selects a number from 1 to 6 and the total is calculated.
- If the total is more than 8 then this process is repeated.
- If the total is 12 (a double six) the player has won.

Outputs

- Displays the title "Dice Game" and the instructions "Throw a double 6 to win. Throw more than 8 and get another try".
- Displays an invalid entry message, if the name is not found.
- Displays a welcome message that includes the player's name.
- Displays dice 1, dice 2 and their combined total each time the dice are thrown.
- Displays the message "Total is over 8. You get another go" if this is the case.
- Displays the message "Double 6 ! You have won" if this is the case.
- Displays the message "Game Over" at the end of the game.

Do not add any additional functionality.

Use comments, white space and layout to make the program easier to read and understand. Save your amended code as **Q07FINISHED.py**

END OF PAPER

Mark Scheme

Question	Answer	Marks	Guidance

1	n declared with value 1. (1)
	Appropriate product input. (1)
	Appropriate size input. (1)
	Product string shortened to first three letters. (1)
	n cast from integer to string. (1)
	product code produced. (1)
	product code output. (1)
		7	n=1
			product=input("Name of product: ")
			size=input("Size(S/M/L)?: ")
			product_code=product[0:3]+ size + str(n)
			print(product_code)

2	fix the syntax error on line 4. (1)
	fix the syntax error on line 6. (1)
	fix the syntax error on line 10. (1)
	change the identifier y to a more meaningful name. (1)
	add a seventh prediction to the game. (1)
	add a comment to explain how the random module is used. (1)
	add two uses of white space to aid readability. (1)
		7	#Edulito practice exams 2 1CP2/02 Q02
			print("Fortune Teller Game")
			name=input("First let's find out who you are, what's your name? ")
			print("ummh",name,"......that is a very interesting name")
			print("now",name,"it's time to read your fortune")
			#random module used so that the prediction is chosen using random number
			import **random**
			x=random.randint(1,6)
			if x==1:
			    print("let me think....it's getting clearer")
			    print(name,"You will have 4 children**")**
			elif x==2:
			    print("yes,yes....it's becoming very clearer")
			    print(name,"you will be a millionaire before the age of 30")
			elif x==3:
			    print("ummmhh....I don't believe it")
			    print(name,"you will be a leader and people will listen carefully to you.")
			elif x==4:
			    print("Can this be true!!!")
			    print(name,"I see you taking part in the Olympics")
			elif x==5:
			    print("can this be!!!")
			    print(name,"you will be happy and healthy and live beyond 100")
			elif x==6:
			    print("what! it is too hazy I must think .....")
			    print(name,"you will have your own TV series")

3 a	Correct indentation. (1)
	Appropriate validation (any method). (1)
	Formula corrected total=total +num (or equivalent). (1)
	num changed to total variable on last line. (1)
		4	reply="Y"
			total=0
			while reply=="Y" or reply=="y":
			    num=int(input ("Enter number: "))
			    total= total + num
			    reply=input("Another?: ")
			    if reply=="N":
			        print ("The total is: ", total)

3 b	Total declared as 0. (1)
	For loop (while loop accepted if used as count-controlled loop). (1)
	Appropriate range. (1)
	Total formula correct. (1)
	Appropriate output of total. (1)
		5	total=0
			for i in range (0,3):
			    num=int(input ("Enter number: "))
			    total= total + num
			print ("The total is: ", total)

4 a	uses a suitable identify for a variable that stores a dog's age. (1)
	inputs from the user the age of the owner's dog using a suitable data type. (1)
	uses a suitable identify for a variable that stores human age equivalent. (1)
	calculates that if the age of the dog equals 1 then the human age equivalent is 12. (1)
	calculates that if the age of the dog is greater than 1 then the human age equivalent is equal to 24 + (4 x dog age - 2). (1)
	outputs the human equivalent age for the owner. (1)
		6	dog_age=int(input("How old is your dog?: "))
			if dog_age==1:
			    human_equiv_age=12
			if dog_age>1:
			    human_equiv_age=24+(4*(dog_age-2))
			print("Human equivalent age: ",human_equiv_age)

4 b	Function defined with correct name (1) and parameter. (1)
	Correct calculations used in body of function. (1)
	Appropriate use of return. (1)
	Appropriate input to main program. (1)
	Function called. (1)
	Appropriate output. (1)
		7	#Function
			def dog_years(age):
			    if dog_age==1:
			        human_equiv_age=12
			    if dog_age>1:
			        human_equiv_age=24+(4*(dog_age-2))
			    return human_equiv_age
			
			#Main Program
			dog_age=int(input(" How old is your dog? "))
			print(dog_years(dog_age))

5 a	Uses for loop or equivalent. (1) Appropriate range. (1) Appropriate use of i or equivalent to output all names. (1) Appropriate use of i or equivalent to output weights in June. (1)	4	for i in range (0,len(weight)): print(weight[i][0],":",weight[i][6])

5 b	Uses for loop or equivalent. (1)
	Appropriate range. (1)
	Calculates weight loss. (1)
	Uses if or equivalent. (1)
	Appropriate comparison. (1)
	Appropriate output. (1)
		6	for i in range (0,len(weight)):
			    weight_loss=weight[i][1]-weight[i][6]
			    if weight_loss>4:
			        print(weight[i][0],":",weight[i][1]-weight[i][6])

6 a	Input for team 1 and 2. (1)
	Input score for team 1 and 2. (1)
	Use of integer for scores. (1)
	If uses appropriate comparison. (1)
	Elif uses appropriate comparison. (1)
	else used appropriately. (1)
	Appropriate output provided. (1)
		6	team1=input("Name of Team 1: ")
			scoreT1=int(input("Score for team 1: "))
			team2=input("Name of Team 2: ")
			scoreT2=int(input("Score for team 2: "))
			if scoreT1>scoreT2:
			    print(team1,"wins!")
			elif scoreT2>scoreT1:
			    print(team2,"wins!")
			else:
			    print("It's a draw!")

6 b	Append statement used within if selection. (1)
	Append statement used to append data to new_entry list. (1)
	Append statement used to append new_entry to the results list.(1)
	New_entry list cleared at the end of each loop. (1)
	Input used appropriately to ascertain whether another match needed to be entered. (1)
	Above repeated for elif.(1)
	Above repeated for else. (1)
	Results list output at end of program. (1)
		8	results=[]
			new_entry=[]
			entry="Y"
			while entry=="Y":
			    team1=input("Name of Team 1: ")
			    scoreT1=int(input("Score for team 1: "))
			    team2=input("Name of Team 2: ")
			    scoreT2=int(input("Score for team 2: "))
			    if scoreT1>scoreT2:
			        print(team1,"wins!")
			        new_entry.append(team1)
			        new_entry.append(3)
			        new_entry.append(team2)
			        new_entry.append(0)
			        results.append(new_entry)
			        new_entry=[]
			        entry=input("Another?: ")
			    elif scoreT2>scoreT1:
			        print(team2,"wins!")
			        new_entry.append(team1)
			        new_entry.append(1)
			        new_entry.append(team2)
			        new_entry.append(3)
			        results.append(new_entry)
			        new_entry=[]
			        entry=input("Another?: ")
			    else:
			        print("It's a draw!")
			        new_entry.append(team1)
			        new_entry.append(1)
			        new_entry.append(team2)
			        new_entry.append(1)
			        results.append(new_entry)
			        new_entry=[]
			        entry=input("Another?: ")
			
			print(results)

7	Program allows name to be input. (1)
	The program accepts lowercase entries of a name and capitalizes the entry to match the list. (1)
	The list **players** is searched to see whether the name entered matches a name in the list. (1)
	Condition controlled loop to allow repeated entry of names.
	Random dice numbers generated and total calculated. (1)
	If the total is more than 8 program repeats. (1)
	
	If the total is 12, appropriate message displayed. (1)
	
	Levels-based mark scheme to a maximum of 9, from:
	• Solution design (3)
	• Good programming practices (3)
	• Functionality (3)
	
	See below
		15	`players=["Jo","Aisha","Dogan","Andy","Lisa","Simon"]`
			`print("Dice Game")`
			`print("throw a double 6 to win.")`
			`print("Throw more than 8 and get another try.")`
			`name=input("Name: ")`
			`name=name.capitalize()`
			`while name not in players:`
			`    print("Invalid entry")`
			`    name=input("Name: ")`
			`else:`
			`    print("Welcome",name)`
			`    import random`
			`    dice1=random.randint(1,6)`
			`    dice2=random.randint(1,6)`
			`    total=dice1+dice2`
			`    print("Dice 1:",dice1,"Dice 2:",dice2,"total: ",total)`
			`    while total>8:`
			`        print("Total is over 8. You get another go ")`
			`        dice1=random.randint(1,6)`
			`        dice2=random.randint(1,6)`
			`        total=dice1+dice2`
			`        print("Dice 1:",dice1,"Dice 2:",dice2,"total: ",total)`
			`        if total==12:`
			`            print("Double 6 ! You have won")`
			
			`    else:`
			`        print("Game over")`

Solution design (levels-based mark scheme)

0	1	2	3	Max
	• There has been little attempt to decompose the problem. • Some of the component parts of the problem can be seen in the solution, although this will not be complete. • Some parts of the logic are clear and appropriate to the problem. • The use of variables and data structures, appropriate to the problem, is limited. • The choice of programming constructs, appropriate to the problem, is limited.	• There has been some attempt to decompose the problem. • Most of the component parts of the problem can be seen in the solution. Most parts of the logic are clear and appropriate to the problem. • The use of variables and data structures is mostly appropriate. • The choice of programming constructs is mostly appropriate to the problem.	• The problem has been decomposed clearly into component parts. • The component parts of the problem can be seen clearly in the solution. • The logic is clear and appropriate to the problem. • The choice of variables and data structures is appropriate to the problem. • The choice of programming constructs is accurate and appropriate to the problem.	3

Good programming practices (levels-based mark scheme)

0	1	2	3	Max
	• There has been little attempt to lay out the code into identifiable sections to aid readability. • Some use of meaningful variable names. • Limited or excessive commenting. • Parts of the code are clear, with limited use of appropriate spacing and indentation.	• There has been some attempt to lay out the code to aid readability, although sections may still be mixed. • Uses mostly meaningful variable names. • Some use of appropriate commenting, although may be excessive. • Code is mostly clear, with some use of appropriate white space to aid readability.	• Layout of code is effective in separating sections, e.g. putting all variables together, putting all subprograms together as appropriate. • Meaningful variable names and subprogram interfaces are used where appropriate. • Effective commenting is used to explain logic of code blocks. • Code is clear, with good use of white space to aid readability.	3

Functionality (levels-based mark scheme)

0	1	2	3	Max
	Functionality (when the code is run) • The component parts of the program are incorrect or incomplete, providing a program of limited functionality that meets some of the given requirements. • Program outputs are of limited accuracy and/or provide limited information. • Program responds predictably to some of the anticipated input. • Solution is not robust and may crash on anticipated or provided input.	Functionality (when the code is run) • The component parts of the program are complete, providing a functional program that meets most of the stated requirements. • Program outputs are mostly accurate and informative. • Program responds predictably to most of the anticipated input. • Solution may not be robust within the constraints of the problem.	Functionality (when the code is run) • The component parts of the program are complete, providing a functional program that fully meets the given requirements. • Program outputs are accurate, informative, and suitable for the user. • Program responds predictably to anticipated input. • Solution is robust within the constraints of the problem.	3

Price List 2020			
Product	Exam Board	Price	
Teacher Pack: Unit 1.1 Systems Architecture J277 (from 2020)	OCR	£ 30.00	
Teacher Pack: Unit 1.2 Memory & Storage J277 (from 2020)	OCR	£ 94.00	
Teacher Pack: Unit 1.3 Computer Networks, Connections and Protocols J277 (from 2020)	OCR	£ 53.00	
Teacher Pack: Unit 1.4 Network Security J277 (from 2020)	OCR	£ 21.00	
Teacher Pack: Unit 1.5 Systems Software J277 (from 2020)	OCR	£ 15.50	
Teacher Pack: Unit 1.6 Ethical, Legal, Cultural and Environmental Impacts of Digital Technology J277 (from 2020)	OCR	£ 21.50	
Teacher Pack: Unit 2.1 Algorithms J277 (from 2020)	OCR	£ 43.00	
Teacher Pack: Unit 2.2 Programming Fundamentals J277 (from 2020)	OCR	£ 74.00	
Teacher Pack: Unit 2.3 Producing Robust Programs J277 (from 2020)	OCR	£ 25.00	
Teacher Pack: Unit 2.4 Boolean Logic J277 (from 2020)	OCR	£ 21.00	
Teacher Pack: Unit 2.5 Programming Languages and IDEs J277 (from 2020)	OCR	£ 15.50	
Teacher Pack Bundle: Component 1 - Computer Systems J277 (from 2020)	OCR	£ 210.00	
Teacher Pack Bundle: Component 2 - Computational Thinking, Algorithms and Programming J277 (from 2020)	OCR	£ 160.00	
Teacher Pack Bundle: Component 1 & 2 - J277 (from 2020)	OCR	£ 350.00	
Topic Test Bundle: Component 1 & 2 (14 tests) - J277 (from 2020)	OCR	£ 28.00	
Homework Pack: Component 1 & 2 (37 homework activities) - J277 (from 2020)	OCR	£ 27.75	
Teacher Pack: Unit 3.1 Fundamentals of Algorithms 8525 (from 2020)	AQA	£ 37.00	
Teacher Pack: Unit 3.2 Programming 8525 (from 2020)	AQA	£ 90.50	
Teacher Pack: Unit 3.3 Fundamentals of Data Representation 8525 (from 2020)	AQA	£ 62.50	
Teacher Pack: Unit 3.4 Computer Systems 8525 (from 2020)	AQA	£ 109.50	
Teacher Pack: Unit 3.5 Fundamentals of Computer Networks 8525 (from 2020)	AQA	£ 34.50	
Teacher Pack: Unit 3.6 Cyber Security 8525 (from 2020)	AQA	£ 27.00	
Teacher Pack: Unit 3.7 Relational Databases and SQL 8525 (from 2020)	AQA	£ 11.50	
Teacher Pack: Unit 3.8 Ethical, Legal and Environmental Impacts of Digital Technology 8525 (from (2020)	AQA	£ 19.00	
Teacher Pack Bundle: AQA GCSE Computer Science 8525 (from 2020) - All Units from 3.1 to 3.8	AQA	£ 340.00	
Topic Test Bundle: Unit 3.1 to 3.8 (13 tests) - 8525 (from 2020)	AQA	£ 26.00	
Homework Pack: Unit 3.1 to 3.8 (34 homework activities) - 8525 (from 2020)	AQA	£ 25.50	
Teacher Pack: Topic 1 & 6 Computational Thinking and Programming 1CP2 (from 2020)	Edexcel	£ 137.00	
Teacher Pack: Topic 2 Data 1CP2 (from 2020)	Edexcel	£ 53.00	
Teacher Pack: Topic 3 Computers 1CP2 (from 2020)	Edexcel	£ 84.50	
Teacher Pack: Topic 4 Networks 1CP2 (from 2020)	Edexcel	£ 60.00	
Teacher Pack: Topic 5 Issues and Impacts 1CP2 (from 2020)	Edexcel	£ 39.50	
Teacher Pack Bundle: Edexcel GCSE Computer Science 1CP2 (from 2020) - All Topics from 1 to 6	Edexcel	£ 330.00	
Topic Test Bundle: Topics 1 to 6 (13 tests) - 1CP2 (from 2020)	Edexcel	£ 26.00	
Homework Pack: Topics 1 to 6 (30 homework activities) - 1CP2 (from 2020)	Edexcel	£ 22.50	

Name & Position			
School & Address			
Email			
Coupon Code:		Order No. (optional)	