1(a) OCR Drones flies goods around the country using drones.

A text file is used to store pilot's codes and date of birth.

The text file is named `pilots.txt`. The contents of some of this file is shown:

```
JP670,17/05/1986
CC200,31/06/1980
SM720,01/04/1976
GC730,06/06/1978
```

Design an algorithm to:

- Ask the user to input a pilot code and a date of birth
- Write these inputs to the `pilots.txt` text file

[4]

(b) A pilot is allowed to fly a maximum of 9 hours per day. The pilot code and the hours flown for each flight in one day are stored in a 2D array of strings with the identifier `journeys`.

The data in `journeys` is shown.

	[0]	[1]
[0]	"SM720"	"4.5"
[1]	"GC730"	"3"
[2]	"JP670"	"2"
[3]	"GC730"	"3.5"
[4]	"CC200"	"9"
[5]	"RY320"	"12"

The value of `journeys[3,1]` is `"3.5"`.

Create a function, `pilotValid()`, that:

- takes a pilot code as a parameter
- uses the 2D array `journeys` to calculate the number of hours that the pilot has flown
- returns `valid` if the pilot has flown 9 hours or fewer or `warning` if the pilot has flown for more than 9 hours
- uses casting where needed.

You must use either:

- OCR Exam Reference Language, or
- A high-level programming language that you have studied.

[6]

2 A player enters a code at the beginning of a computer game. The code is three characters long. The code sets the level of the game. The game has three levels.

A variable `level` is assigned a value based on the code entered.

i. Write an algorithm that:

- asks the user to enter a three-character code
- repeats the previous step if the code is not 3 characters long
- if the code entered is SVA, assigns the value 2 to `level`
- if the code entered is UTV, assigns the value 3 to `level`
- assigns the value 1 to `level` if any other three-character code is entered.

[6]

ii. When a player completes a level, they move to the next level.

Once level 3 is completed, the game starts again at level 1.

Write a function called `nextlevel()` that:

- takes the current level as a parameter
- works out the integer value of the next level the player moves to

- returns the integer value.

[3]

iii. Complete the line of code to output the value returned from the function `nextlevel()` when the integer value 3 is passed in as a parameter.

```
print(……..……...…..……..…..(……..…....……..….))
```

[1]

3(a) OCR Drones flies goods around the country using drones.

Details about the drones that pilots fly are stored in a database table called `TblDrone`.
Some of the data stored in this table is shown.

DroneID	DroneType	Mileage	LastCheck
001	Quadcopter	65 032	65 000
002	Quadcopter	32 128	21 000
003	Octocopter	98 021	98 000

TblDrone

i. Complete the SQL statement to display `DroneID` and `Mileage` for all Octocopter type drones that have a mileage of greater than 50 000 miles.

```
SELECT ……

…… TblDrone

WHERE DroneType = "Octocopter" ……..…… Mileage ……..……..……
```

[4]

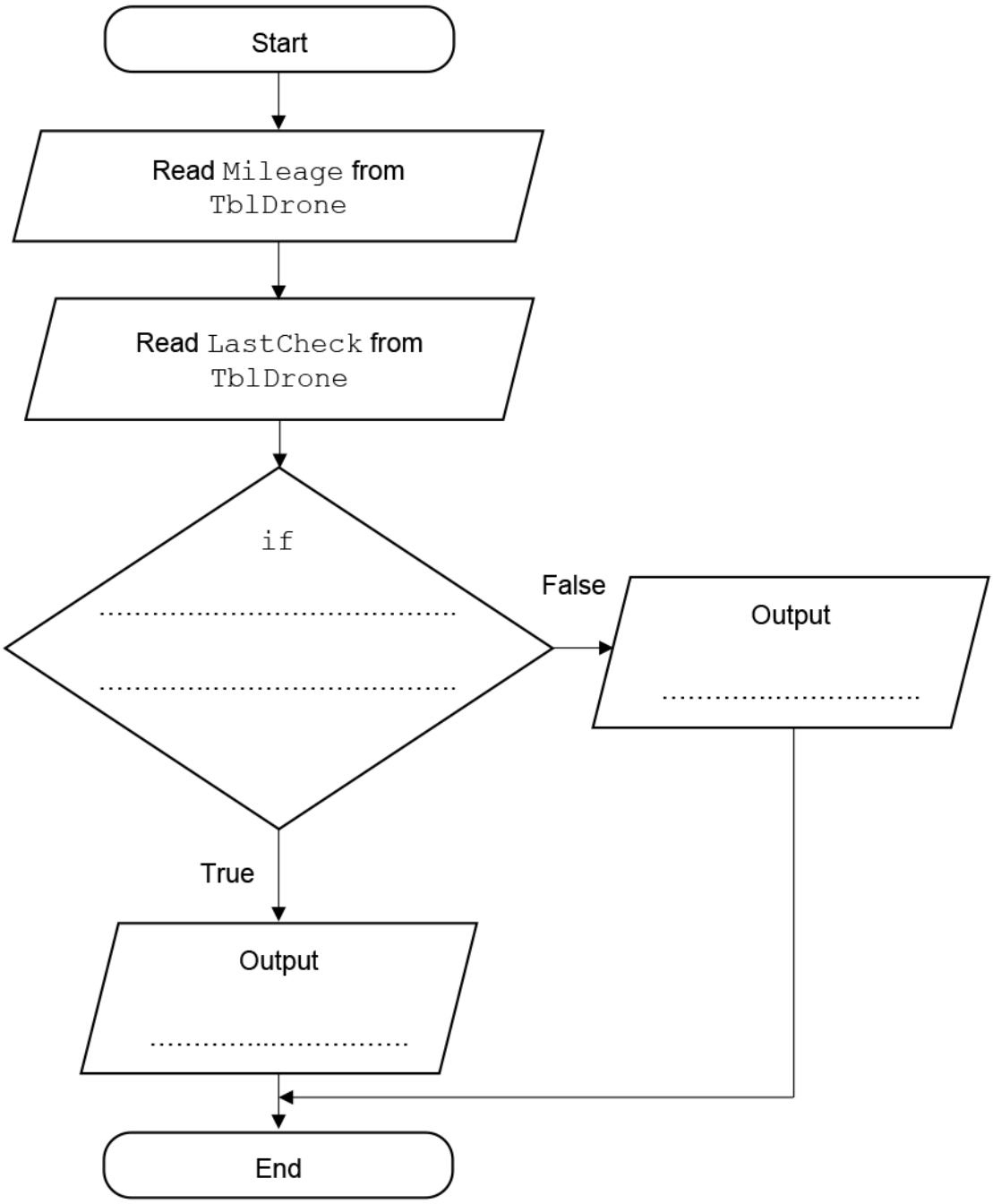
ii. Drones must be checked every 10 000 miles. If the difference between `Mileage` and `LastCheck` is greater than 10 000 then the drone needs to be checked.

A flowchart shows the steps needed to check a drone.

The flowchart outputs `"Check"` if the drone needs to be checked.

The flowchart outputs `"No Check"` if the drone does **not** need to be checked.

Complete the flowchart for the algorithm.


Start
Read Mileage from TblDrone
Read LastCheck from TblDrone
if
.....
.....
False
Output
.....
True
Output
.....
End

[2]

(b) A pilot code is automatically generated when a new pilot joins the company.

This algorithm generates a code for each pilot:

```
01 a = input("Enter first letter of first name")
02 b = input("Enter first letter of second name")
03 c = random(1,100)
04 while c < 100
05 c = c * 10
06 endwhile
07 pilotCode = a + b + str(c)
08 print(pilotCode)
```

Complete the trace table for the given algorithm.

Lines 01 to 03 have already been completed.

You may not need to use all rows in the trace table.

Line number	a	b	c	pilotCode	Output
01	H				
02		K			
03			9		

[4]

(c) Pilots are paid a set amount each day. Pilots also get an additional payment for each mile they have flown that day. These payments are shown in the table.

Flying experience	Pay per day	Pay per mile
Fewer than 2 years	£120.00	£0.45
2 years to 5 years inclusive	£150.00	£0.65
More than 5 years	£180.00	£0.85

For example, a pilot with 3 years' experience who flies 100 miles in one day will receive a total of £215.00 pay. This is calculated in the following way:

- £150.00 pay for the day
- £65.00 additional payment (£0.65 pay per mile x 100 miles)
- Add £150.00 and £65.00 together to get £215.00 total pay

i. Complete the algorithm to:

- Calculate the total pay for the pilot for that day

You must use either:

- OCR Exam Reference Language, or
- A high-level programming language that you have studied.

```
experience = input("Enter years of experience")

miles = input("Enter miles flown")

totalPay = 0

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

print(totalPay)
```

[4]

ii. The programmer decides to make a function to calculate the total pay for the pilot.
The function is called `calculatePay()`.

The function takes the values for `experience` and `miles` as two parameters and returns the total pay for the pilot.

Refine the algorithm to use this function and output the pay for the pilot.

You must use either:

- OCR Exam Reference Language, or
- A high-level programming language that you have studied.

```
experience = input("Enter years of experience")

miles = input("Enter miles flown")


…… = calculatePay( …… , ……)


print( totalPay )
```

[2]

4(a) An alarm has an algorithm that decides whether to sound the alarm by checking the data that is stored in the following three variables.

- `SystemArmed`
- `DoorSensorActive`
- `WindowSensorActive`

The alarm will only sound when the alarm has been activated **and** one or both of the door and window sensors are activated. When the system needs to sound the alarm it calls the pre-written procedure `SoundAlarm()`

Write a program that checks the data in the variables and calls `SoundAlarm()` when appropriate.

You must use **either**:

- OCR Exam Reference Language, **or**
- A high-level programming language that you have studied.

---

[4]

(b) An alarm system has a log that stores a record each time a sensor is triggered. This is called an event. The record format is given in the table:

Fieldname	Description
`Date`	The date the event happened
`SensorID`	The sensor that was activated
`SensorType`	The type of sensor that was activated – Door, Motion or Window
`Length`	The number of seconds the sensor was triggered (to the nearest second)

The log is stored in a database table called `events`. The current contents of `events` is shown:

Date	SensorID	SensorType	Length
05/02/2023	WS2	Window	38
05/02/2023	MS1	Motion	2
06/02/2023	DS3	Door	1
06/02/2023	MS2	Motion	3
06/02/2023	MS1	Motion	2
07/02/2023	WS1	Window	24
07/02/2023	DS1	Door	1

Write an SQL statement to display the sensor IDs of the door sensors that have been triggered for more than 20 seconds.

---

[3]

(c) A program written in a high-level language is used to access the data from a database.
This program has a procedure, `SaveLogs()`, that stores the data to an external text file.

The procedure `SaveLogs()`:

- takes the string of data to be stored to the text file as a parameter
- takes the filename of the text file as a parameter
- stores the string of data to the text file.

Write the procedure `SaveLogs()`

You must use **either**:

- OCR Exam Reference Language, **or**
- A high-level programming language that you have studied.

[6]

(d) OCR Security Services need to identify the total number of seconds the sensors have been activated on a specific date.

The data from the database table `events` is imported into the program written in a highlevel programming language.

The program stores the data in a two-dimensional (2D) string array with the identifier `arrayEvents`

The data to be stored is shown in the table.

Date	SensorID	SensorType	Length
05/02/2023	WS2	Window	38
05/02/2023	MS1	Motion	2
06/02/2023	DS3	Door	1
06/02/2023	MS2	Motion	3
06/02/2023	MS1	Motion	2
07/02/2023	WS1	Window	24
07/02/2023	DS1	Door	1

In this table, the value of `events[1, 1]` contains "`MS1`".

i. An array can only store data of one data type. Any non-string data must be converted to a string before storing in the array.

Identify the process that converts integer data to string data.

---

[1]

ii. Write a program that:

- asks the user to input a date
- totals the number of seconds sensors have been activated on the date input
- outputs the calculated total in an appropriate message including the date, for example:
  `Sensors were activated for 40 seconds on 05/02/2023`

You must use **either**:

- OCR Exam Reference Language, **or**
- A high-level programming language that you have studied.

---

[6]

5(a) Customers at a hotel can stay between 1 and 5 (inclusive) nights and can choose between a basic room or a premium room.

A typical booking record is shown in the table:

`firstName`	Amaya
`surname`	Taylor-Ling
`nights`	3
`room`	Premium
`stayComplete`	False

i. State the most appropriate data type for the following fields:

Nights

Room

[2]

ii. Give the name of **one** field that could be stored as a Boolean data type.

[1]

iii. Booking records are stored in a database table called `TblBookings`.

The following SQL statement is written to display all customer bookings that stay more than one night.

```
SELECT ALL

FROM TblBookings

IF Nights < 1
```

The SQL statement is incorrect.

Rewrite the SQL statement so that it is correct.

[4]

(b) When a new booking is recorded, the details are entered into a program to validate the values. The following criteria are checked:

- `firstName` and `surname` are not empty
- `room` is either "basic" or "premium"
- `nights` is between 1 and 5 (inclusive).

If any invalid data is found "NOT ALLOWED" is displayed.

If all data is valid "ALLOWED" is displayed.

i. Complete the following program to validate the inputs.

You must use **either**:

- OCR Exam Reference Language, **or**
- a high-level programming language that you have studied.

```
firstName = input("Enter a first name")

surname = input("Enter a surname")

room = input("Enter basic or premium")

nights = input("Enter between 1 and 5 nights")

stayComplete = False
```

[5]

ii. Complete the following test plan to check whether the number of nights is validated correctly.

Test data (number of nights)	Type of test	Expected output
2		ALLOWED
	Boundary	ALLOWED
	Erroneous / Invalid	NOT ALLOWED

[3]

(c) A Basic room costs £60 each night. A Premium room costs £80 each night.

i. Create a function, `newPrice()`, that takes the number of nights and the type of room as parameters, calculates and returns the price to pay.

You do **not** have to validate these parameters.

You must use **either**:

- OCR Exam Reference Language, **or**
- a high-level programming language that you have studied.

[4]

ii. Write program code, that uses `newPrice()`, to output the price of staying in a Premium room for 5 nights.

You must use **either**:

- OCR Exam Reference Language, **or**
- a high-level programming language that you have studied

[3]

(d) A hotel car park charges £4 per hour. If the car is electric, this price is halved to £2 per hour.

Write an algorithm to:

- take as input the number of hours the user has parked and whether their car is electric or not
- calculate and output the total price
- repeat continually until the user enters 0 hours.

You must use **either**:

- OCR Exam Reference Language, **or**
- a high level programming language that you have studied.

[6]

6 The database table `Results` stores the results for each student in each of their chosen subjects.

StudentName	Subject	Grade
Alistair	English	3
Jaxon	Art	5
Alex	Art	4
Anna	French	7
Ismaael	Art	9

Complete the SQL query to return all of the fields for the students who take Art.

```
SELECT
FROM
WHERE
```

[3]

7 A cinema uses the following criteria to decide if a customer is allowed to see a film that has a 15 rating:

Customers have to be 15 years of age or older to see the film. They also need to either have a ticket or have the money to buy a ticket.

The table shows the inputs to the system that will output whether the customer can watch the film.

Input	Criteria (True / False)
A	The customer is 15 or over
B	The customer has a ticket
C	The customer has the money to buy a ticket

The cinema has three screens: "Red", "Black" and "Yellow".

The function `freeseats()` counts how many seats are available in each screen. The name of the screen is passed in as a string parameter and the number of free seats is returned as an integer.

Write code using the function `freeseats()` to find the number of seats available in screen Red and assign this to a variable with identifier `redseats`.

---

[2]

8(a) OCR Tech is an online shop that sells electronics such as TVs and game consoles.

Items for sale are stored in the database table `tblStock`. An extract of this table is shown.

ItemCode	ItemName	Price (£)	Stock
GSC5	GameStation5 console	249.99	102
TV4K	4K Television	499.99	18
ABRR	Audiobook reader	59.99	27
NAGC	TV streaming stick	24.99	192

`tblStock`

Tick **(✓) one** box in each section to identify the correct SQL statement to select the item code and item name for all items that have a price of £60 or over.

	Tick (✓) **one** box
`SELECT ItemCode AND ItemName`	
`SELECT ItemCode, ItemName`	
`SELECT ItemCode & ItemName`	

	Tick (✓) **one** box
`FROM tblStock`	
`FROM table`	
`FROM database`	

	Tick (✓) **one** box
`WHERE Price <= 60`	
`WHERE Price > 60`	
`WHERE Price >= 60`	

[3]

(b) Customers can use a discount code to reduce the price of their purchase. Valid discount codes and their value (in pounds) are stored in a global two-dimensional (2D) array with the identifier `discount`. The following table shows part of this 2D array.

	0	1
0	PVFC7	10
1	CPU5	5
2	BGF2	15

For example, `discount[2,0]` holds discount code BGF2 and `discount[2,1]` holds the discount of 15 pounds.

A function searches through the 2D array and applies the discount to the price. The price and discount code are passed in as parameters. The algorithm design is not complete.

i. Complete the design for the algorithm.

```
function checkdiscount(price, code)
    newprice = price
    size = len(discount) - 1
    for x = 0 to ....................
      if discount[x,0] == ................ then
        newprice = ................ - discount[................]
      endif
    next x
    ............................
endfunction
```

[5]

ii. Identify **two** variables used in this function design.

1 _____

2 _____

[2]

iii. Write a program that:

- asks the user for an item price and discount code
- uses the `checkdiscount()` function from **part (i)** to calculate the price of the item after any discount has been applied
- repeats bullet points 1 and 2 until a price of 0 is entered
- outputs the total cost of all items entered, after any discounts have been applied.

You must use **either:**

- OCR Exam Reference Language, **or**
- A high-level programming language that you have studied

[6]

9 Data for one week (Monday to Friday) is stored in a 2D array with the identifier `minsPlayed`.

The following table shows part of this array, containing 4 students.

			Students			
			Stuart	**Wes**	**Victoria**	**Dan**
			0	**1**	**2**	**3**
Days of the week	**Mon**	**0**	60	30	45	0
	Tue	**1**	180	60	0	60
	Wed	**2**	200	30	0	20
	Thu	**3**	60	10	15	15
	Fri	**4**	100	35	30	45

The teacher wants to output the number of minutes Dan (column index 3) played computer games on Wednesday (row index 2). The following code is written:

```
print(minsPlayed[3,2])
```

Write program code to output the number of minutes that Stuart played computer games on Friday.

You must use **either**:

- OCR Exam Reference Language, **or**
- a high-level programming language that you have studied.

[1]

END OF QUESTION PAPER

<table>
<tr><th colspan="3">Question</th><th>Answer/Indicative content</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>1</td><td>a</td><td></td><td>One mark per bullet point

• Ask the user for two inputs and store/use these
• Open pilots.txt (for write/append)
• Write both inputs to opened text file
• Close text file</td><td>4
(AO3 2b)</td><td>Award BP4 for implicit closing of file (e.g. using <code>with…</code> in Python)

Example answer :
<pre>dronepilotData =
open("pilots.txt")
pilotCode = input("enter code")
dob = input("enter date of
birth")
dronepilotData.writeLine(pilotC
ode, dob)
dronepilotData.close()</pre>
Allow data to be written either by simply writing both values or by concatenating into one string with a separating comma.</td></tr>
<tr><td></td><td>b</td><td></td><td>One mark per bullet point to max 6

• Function header <code>pilotValid()</code> with (at least one) parameter
• Checks array element [i,0] for each item…
• …calculates total
• …casting to float / real
• Checks if 9 hours or fewer
• Returns a value
• …returns "warning" and "valid" correctly</td><td>6
(AO3 2b)</td><td>Example answer :
<pre>function pilotValid(pilotCode)
  total = 0
  status = ""
  for i = 0 to 5
      if journeys[i,0] ==
pilotCode then
        temp =
float(journeys[i,1])
        total = total + temp
      endif
  next i
  if total > 9 then
      status = "warning"
  else
      status = "valid"
  endif
  return status
endfunction</pre>
Do not allow casting to integer for BP4, data shows some journeys have decimal places.

Allow FT for BP5 if attempt made at calculation.
Allow FT for BP7 if value is output instead of returned.</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>10</td><td></td></tr>
</table>

<table>
<tr><th colspan="3">Question</th><th>Answer/Indicative content</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>2</td><td></td><td>i</td><td>1 mark per bullet point

• Input code from user <u>and store/use</u>
• Repeat for non-three character codes
• Check for one code…
• …and set level appropriately
• Check for and set level for 2nd code
• Set level to 1 for any other 3 character code</td><td>6
(AO3 2b)</td><td>Example answer :
<pre>level = 0
code = ""
while code.length != 3 then
   code = input("enter a 3
character code")
endwhile

switch code :
   case "SVA":
     level = 2
   case "UTV":
     level = 3
   default:
     level = 1
endswitch</pre>alternative example answer
<pre>level = 0
code = ""

while code.length != 3 then
     code = input("enter a 3
character code")
endwhile
if code == "SVA" then
     level = 2
elseif code == "UTV" then
     level = 3
else
     level = 1
endif</pre></td></tr>
<tr><td></td><td></td><td>ii</td><td>1 mark per bullet point

• function <code>nextlevel()</code> defined with <u>at least</u> one parameter
• returns parameter plus 1…
• … except for if parameter is 3, then returns 1</td><td>3
(AO3 2b)</td><td>Example answer :
<pre>function nextlevel(oldlevel)
    if oldlevel == 3 then
        return 1
    else
        return oldlevel + 1
    endif

end function</pre>Accept correct use of MOD for BP2 and 3.

Allow FT for BP3 if value output / calculated but not returned.</td></tr>
<tr><td></td><td></td><td>iii</td><td>• print(nextlevel(3))</td><td>1
(AO3 2b)</td><td>Allow other logically correct answers</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>10</td><td></td></tr>
</table>

<table>
<tr><th colspan="3">Question</th><th>Answer/Indicative content</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>3</td><td>a</td><td>i</td><td>• DroneID, Mileage
• FROM
• AND
• > 50000</td><td>4
(AO3 2b)</td><td>Accept SELECT * or selection of additional fields for BP1.</td></tr>
<tr><td></td><td></td><td>ii</td><td>• <code>if Mileage-LastCheck > 10000</code>
• Output "Check" and "No Check" or equivalent correctly based on logical check for BP1</td><td>2
(AO3 2a)</td><td>BP1 could be >, < or either in words.

Ignore case and minor misspellings.

BP2 (output) could be either way around depending on comparison for BP1.</td></tr>
<tr><td></td><td>b</td><td></td><td>1 mark per row

• c = 90 on line 05
• c = 900 on line 05
• pilotCode = HK900 on line 07
• HK900 output on line 08</td><td>4
(AO3 1)</td><td>Ignore additional lines that do not affect outcome.
FT for missing or incorrect line numbers.
FT for output based on incorrect tracing of loop.

<table><tr><th>Line number</th><th>a</th><th>b</th><th>c</th><th>pilotCode</th><th>Output</th></tr><tr><td>01</td><td>H</td><td></td><td></td><td></td><td></td></tr><tr><td>02</td><td></td><td>K</td><td></td><td></td><td></td></tr><tr><td>03</td><td></td><td></td><td>9</td><td></td><td></td></tr><tr><td>05</td><td></td><td></td><td>90</td><td></td><td></td></tr><tr><td>05</td><td></td><td></td><td>900</td><td></td><td></td></tr><tr><td>07</td><td></td><td></td><td></td><td>HK900</td><td></td></tr><tr><td>08</td><td></td><td></td><td></td><td></td><td>HK900</td></tr></table></td></tr>
<tr><td></td><td>c</td><td>i</td><td>• Attempt at using selection
• Calculates pay correctly for pilots with less than 2 years experience
• Calculates pay correctly for pilots with 2 to 5 years experience. Must include both 2 and 5.
• Calculates pay correctly for pilots with more than 5 years experience</td><td>4
(AO3 2b)</td><td>Example answer :

<pre>experience = input(“Enter years of experience”)
miles = input(“Enter miles flown”)

totalPay = 0
if exp <2 then
    totalPay = 120+(0.45*miles)
elseif exp <=5 then
    totalPay = 150+(0.65*miles)
else
    totalPay = 180+(0.85*miles)
end if
print(totalPay)</pre>
FT for BP4 only where a reasonable attempt at calculating pay has been made.</td></tr>
<tr><td></td><td></td><td>ii</td><td>• totalPay
• (experience, miles) // (miles, experience)</td><td>2
(AO3 2c)</td><td><code>totalPay = calculatePay(experience, miles)</code></td></tr>
</table>

Question			Answer/Indicative content	Marks	Guidance
			Total	16	

Question			Answer/Indicative content	Marks	Guidance

4	a		1 mark each:
			
			• Attempt at using selection / condition controlled loop
			• Checking if system armed // while system armed
			• If Door Sensor active **OR** Window Sensor active (both checks required)
			• calling SoundAlarm **correctly**
				4 (AO3 2b)	Selection could be done using IF statement, case statement or any other sensible valid method.
					
					Allow reference to `AlarmActivated` or equivalent instead of `SystemArmed`
					
					Ignore any inputs or modification of variables.
					
					Allow True / False as strings. Allow checking against strings (e.g. if `SystemArmed == "active"`)
					
					Allow checking armed/disarmed for BP2 and BP3
					
					Only award BP4 if SoundAlarm correctly called / not called in every situation. If issues on previous lines (e.g. lack of brackets where needed) means this is not the case, do not award BP4.
					
					Checking could be done by evaluating variable directly (`if SystemArmed`) or by comparison (`if SystemArmed == True`)
					
					<u>Example answer 1</u>
					
					`if SystemArmed then`
					`  if DoorSensorActive then`
					`  SoundAlarm()`
					`  else if WindowSensorActive then`
					`  SoundAlarm()`
					`  endif`
					`endif`
					
					<u>Example answer 2</u>
					
					`while SystemArmed then`
					`  if DoorSensorActive then`
					`  SoundAlarm()`
					`  else if WindowSensorActive then`
					`  SoundAlarm()`
					`  endif`
					`endif`
					
					<u>Example answer 3</u>
					
					`if SystemArmed and (DoorSensorArmed or`

Question	Answer/Indicative content	Marks	Guidance

			WindowSensor) then
			  SoundAlarm()
			endif
			
			Note – above example needs brackets,
			
			if SystemArmed and DoorSensorArmed or WindowSensor then
			
			is not logically valid for this scenario (will sound alarm when not armed if window sensor is active)
			
			<u>Example answer 4</u>
			
			if SystemArmed and DoorSensorArmed
			  SoundAlarm()
			else if SystemArmed and WindowSensorArmed
			  SoundAlarm()
			endif
			
			<u>Examiner's Comments</u>
			
			Many responses achieved highly on this question. The question asks for a simple program to be written that checks the given variables and calls the given procedure when necessary.
			
			Examiners were instructed to be generous with the first mark, crediting any use of selection or condition-controlled iteration. Responses may therefore have been rewarded for an attempt at this question even if their solution was not fully functional.
			
			Centres should encourage candidates to attempt each question for precisely this reason; it is typical that a small number of marks are allocated to attempting a solution on many programming questions for J277/02.
			
			A significant number of responses were given 3 out of 4 marks as they misunderstood the role of operator precedence in their solution; this is detailed in the "misconception" box below.
			
			Misconception

Question	Answer/Indicative content	Marks	Guidance

			
			Where multiple conditions are used in selection, these have an order of precedence very much like BIDMAS does in mathematics; an `AND` operator will always take precedence over an `OR` operator. A `NOT` operator (not used in this question) would have even higher precedence.
			
			This can cause problems in candidate responses. A common candidate response was:
			
			`if SystemArmed` **`AND`** `DoorSensorActive` **`OR`** `WindowSensorActive then SoundAlarm()`
			
			However, because the `AND` operator takes precedence, the first check done here is if the system is armed and the door sensor is active. The result of this is then evaluated with an OR operator to check if the window sensor is active.
			
			This results in the alarm sounding if the window sensor is active, even if the system is not armed. This was clearly not the candidate's intention.
			
			To fix this, candidates could have either:
			
			• put brackets/parentheses around the `Door OR Window` section of their response
			• written the response as separate checks. This could have been done in multiple ways, including nested `if` statements or repeated checks.
			
			Exemplar 1
			
			`if SystemArmed:`
			`  if DoorSensorActive or WindowSensorActive:`
			`  SoundAlarm()`
			
			Exemplar 1 shows one way that full marks are achieved on this question. The candidate has used nested `if` statements to check if the system is armed, and if

Question			Answer/Indicative content	Marks	Guidance
					true, then checking if either sensor has been activated. The `SoundAlarm()` procedure is only called if both if statements evaluate to True.

	b		1 mark each
			
			• `SELECT SensorID // SELECT *`
			• `FROM events`
			• `WHERE Length > 20 AND sensorType = “Door”`
			`//`
			`WHERE sensorType = “Door” AND Length > 20`
				3
				(AO3 2c)
					Max 2 if out of order or anything extra that affects the output.
					
					BP1 can select multiple fields as long as `SensorID` is included.
					
					Ignore case. Only penalise spaces if obvious.
					
					Field names must be correct.
					
					“door” must be in quotation marks for BP3. Allow quotation marks for field names and table name
					
					BP3 can use == or = for equivalence.
					
					Allow alternative WHERE clauses that are logically correct (e.g. `WHERE length >=21`)
					
					<u>Examiner’s Comments</u>
					
					Structured Query Language is obviously well understood by many candidates. Many high-quality responses were produced.
					
					Most responses were able to use SELECT and FROM appropriately to produce a logically correct response. However, the vast majority of responses missed off the requirement that only door sensors were required to be included, gaining 2 out of 3 marks in the process.
					
					Although a suggested response is shown in the mark scheme, any logically correct SQL that produces the required output would be accepted by examiners.
					
					Where a mistake was made consistently (such as using colons after the SQL keyword), this was penalised once and then FT (follow through) allowed for subsequent marks.

Question	Answer/Indicative content	Marks	Guidance

c	1 mark each
	
	• **Define** procedure SaveLogs…
	• …with **two** valid parameters
	• Open file (for write/append) …
	• … using the file name **passed in as parameter**
	• Write data **to file**…
	• …using the data **passed in as parameter**
	• Close file
		6
		(AO3 2b)
			Must be clear that answer is a procedure definition, do not credit calling procedure for BP1. Allow function definition.
			
			If parameters are later overwritten, do not credit BP2 but FT for BP4 and 6.
			
			Closing text file does not need reference to file name/object – e.g. "`close file`" is enough. However, if given reference must be correct.
			
			If code given outside of procedure, do not give BP4 and BP6
			
			Allow FT for multiple occurrences of same mistake (e.g. not using filename correctly for open and close)
			
			<u>Example answer</u>
			
			`procedure SaveLogs(data, filename)`
			`  logFile = open(filename)`
			`  logFile.writeLine(data)`
			`  logFile.close()`
			`endprocedure`
			
			<u>Examiner's Comments</u>
			
			This question proved to be challenging for many candidates. The question combined defining a procedure with the use of text files.
			
			The tasks required were partially decomposed in the bullet points. A candidate attempting these in order would have achieved a significant number of marks.
			
			Candidates could also have achieved numerous marks for a partial solution (e.g. defining a procedure that didn't use text files or writing to a text file outside of a procedure) and the mark scheme was deliberately constructed to credit these responses.
			
			Full marks were often given where candidates appear to have had practical experience of these two techniques.
			
			Exemplar 2

Question			Answer/Indicative content	Marks	Guidance
					procedure SaveLogs(data, fileName) file = open(fileName) file.writeLine(data) file.close() endprocedure Exemplar 2 shows a response that scored full marks. The procedure has been defined with multiple parameters which are then used to open the file and to write the data. The candidate has also achieved the bullet point 7 on the mark scheme (closing the file) but this wasn't necessary in this case.
	d	i	1 mark for: • Casting / cast	1 (AO3 2a)	Accept type casting Do not accept conversion. Do not accept examples of casting. Examiner's Comments The use of the term "casting" to convert one data type to another is now well known and understood by candidates. This is given and referred to in the J277 specification and is essential knowledge.
		ii	1 mark each to max 6 • Input date and store in variable / use directly • Access **all** seven (indexes 0 to 6) events in array // loop for each event in array • Attempt at selection… • …to compare `date` input against `date` in array (element 0) • …adding `length` (element 3) from array to the total if dates match. • Outputting calculated total and date in appropriate message(s) at the end	6 (AO3 2b)	BP2 can be achieved either by iteration accessing each event or manually repeating code to access each event. Must be 0 to 6, not 1 to 7. Allow reference `to events` (table given) or `arrayEvents` (2D array) in answer as long as used consistently. BP2 loop allow off by one errors (Python), looping to array length or array length – 1. Allow `for each item` in array or any other suitable loop. BP4 and BP5 allow array reference as either column major or row major. Output can either be once at the end or on every iteration, as long as it is output at the end. Only give output mark if attempt made to **calculate** total within the algorithm. Do not penalise capitalisation or minor misspellings of variable names. Example answer 1

Question	Answer/Indicative content	Marks	Guidance

			total = 0
			date = input("Please enter date")
			for count = 0 to events.length-1
			  if events[0, count] == date then
			  total = total + events[3,count]
			  endif
			next count
			print("There were " + total + " events on " + date)
			
			Example answer 2
			
			total = 0
			date = input("Please enter date")
			for item in events:
			  if item[0] == date then
			  total = total + item[3]
			  endif
			next count
			print("There were " + total + " events on " + date)
			
			Examiner's Comments
			
			The final question in Section B is expected to be a high demand question.
			
			The techniques required (iteration through a 2D array, selection, keeping a running total of times) are within the specification but it is acknowledged that the level of challenge was high.
			
			Examiners were instructed to give marks for an attempt at a solution (as with previous questions).
			
			For this question marks were given for:
			
			• any attempt at selection
			• any solution that accessed each element in the given array, even if this was via a manual process.
			
			Therefore, many candidates gained multiple marks for an attempt that only partially solved the problem.
			
			A significant number of candidates were able to create a solution that fully met the

Question	Answer/Indicative content	Marks	Guidance

			requirements of the question. This was often done in an elegant and efficient manner.
			
			This is extremely pleasing and shows excellent understanding and significant experience of practical programming.
			
			Exemplar 3
			
			total=0
			date = input ("Enter a date")
			~~toto~~
			~~count = 0~~
			for count = 0 to arrayEvents.length
			if arrayEvents[0, count] == date then
			total = total + arrayEvents[3, count]
			endif
			~~endfor~~
			
			print ("Sensors were activated for", total, "seconds on", date)
			
			Exemplar 3 shows a high scoring response. A date has been asked for as input which has then been used to compare to each element at position 0 in the array.
			
			Where any of these match, the total variable is updated to keep a running total of the corresponding element at position 3 in the array.
			
			After each element has been checked, the total and date are output in a suitable message.
			
			This is not the only method by which a response could be given full marks but is perhaps the most common.

	Total	20	

Question			Answer/Indicative content	Marks	Guidance
5	a	i	• Integer • String	2 (AO3 2a)	Accept other valid data types from high-level languages (e.g. byte, short for integers) Do not accept descriptions (e.g. “whole number”, “text”). Do not accept “character(s)” for string.
		ii	• stayComplete	1 (AO3 2a)	Ignore spaces or misspelling as long as recognisable. <u>Examiner’s Comments</u> These questions tested candidates’ knowledge of data types and it was clear that this knowledge was well understood. The majority of candidates were able to correctly identify suitable data types in section (i) and identify `stayComplete` as the field that would be stored as a Boolean data type.

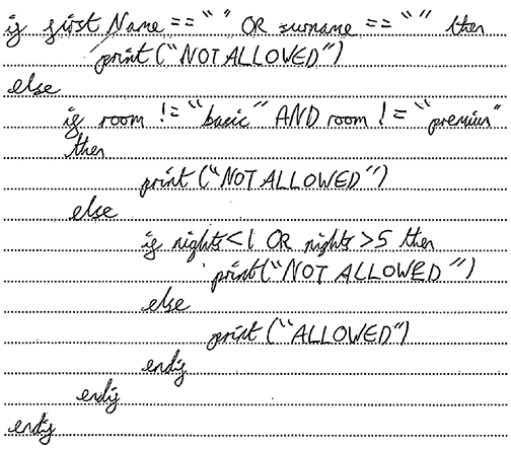
Question			Answer/Indicative content	Marks	Guidance
		iii	• `SELECT FirstName, Surname, Nights, Room, StayComplete // SELECT*` • `FROM TblBookings` • `WHERE` • `Nights > 1 // Nights >= 2 // Nights BETWEEN 2 AND 5.`	4 (AO3 1) (AO3 2c)	Order of fields for BP1 not important but must show all fields and be separated by commas. Ignore capitalisation and spacing. Spelling must be correct. Ignore quotes around numeric values or field/table names. Allow other logically valid SQL statements. Check with TL if required. Ignore reference to `stayComplete` or other valid SQL code that would not affect output. Max 3 if in wrong order or if includes any extra invalid code <u>Examiner's Comments</u> This question tested candidates' ability to refine and rewrite incorrect code given to them. It is important to note that although the SQL statement as a whole is incorrect, not all components are incorrect; in this case, the `FROM` clause is correct and candidates who made no change to this line were credited. `SELECT ALL` is invalid SQL and should have been written to instead include all fields from the table, separated by commas `SELECT *` was equally accepted as a suitable response. `IF` is not a valid SQL keyword and needs to be replaced with `WHERE`. The criteria for this statement was also incorrect. The comparison symbol is incorrect and should read `Nights > 1`. Most candidates gained some marks on this question. The most common response corrected the criteria and not modifying the `FROM` clause.

Question			Answer/Indicative content	Marks	Guidance

	b	i	• Checks that **both** `firstname` and `surname` are not empty…
			• Checks that `room` is either “basic” or “premium”…
			• Checks that `nights` is between 1 and 5 (inclusive)…
			• …Outputs “NOT ALLOWED” (or equivalent) if <u>any</u> of the 3 checks are invalid (must check all three)
			• …Outputs “ALLOWED” (or equivalent) <u>only</u> if all three checks are valid (must check all three)
			
			Note : output marks are given for if entire system produces the correct output. For example, If a user enters a valid name and room but an invalid number of nights, the system should say “NOT ALLOWED” (or equivalent). If this works and produces the correct response no matter which input is invalid, BP4 should be given.
			
			The same process holds for the valid output - if (and only if) three valid inputs results in an output saying “ALLOWED” (or equivalent), BP5 should be given. Do not give this if ALLOWED is printed when (for example) two inputs are valid and one is invalid.
			
			For any output marks to be given, a sensible attempt must have been made at all three checks. These may not be completely correct (and may have been penalised in BPs 1 to 3) but should be enough to allow the FT marks for output.
				5
				(AO3 2a)
					Must have some attempt at <u>all three checks</u> to give output mark(s). Check for `nights` must check both upper and lower limits.
					
					Iteration can be used as validation if input repeatedly asked for until valid answer given.
					
					<u>Do not accept</u> logically incorrect Boolean conditions such as `if firstname or surname == “”`
					
					Do **not** accept ≥ or ≤ for >=, <=. Ignore capitalisation
					
					e.g.
					
					`valid = True`
					`if firstname == “” `**<u>or</u>**` surname == “” then`
					`  valid = False`
					`end if`
					`if room != “basic” `**<u>and</u>**` room != "premium" then`
					`  valid = False`
					`endif`
					`if nights < 1 `**<u>or</u>**` nights > 5 then`
					`valid = False endif`
					`  if valid then`
					`print(“ALLOWED”)`
					`else`
					`  print(“NOT ALLOWED”)`
					`endif`
					
					BP1 to 3 can check for valid or invalid inputs. . Pay particular attention to use of AND / OR. Only give marks for output if these work together correctly.
					
					Example above shows checking for **invalid** data. Checks for **valid** data equally acceptable Examples shown below:
					
					• `if firstname != “”` <u>and</u> `surname != “”`
					• `if room == “basic”` <u>or</u> `room == “premium”`
					• `if nights >= 1` **<u>and</u>** `nights <= 5`
					
					<u>Examiner’s Comments</u>
					
					This question stretched the understanding of even highly-achieving candidates and it

Question	Answer/Indicative content	Marks	Guidance
			was not uncommon to see low scoring responses. Misunderstanding of Boolean operators (AND and OR) within selection (IF) statements was something that affected a lot of candidate responses. As this question was in Section B, candidates needed to respond in OCR Exam Reference Language or a high-level language. Responses must be logically correct to gain the marks. As each check is two individual checks that both need to pass, responses can quickly get relatively complicated. As can be seen from the mark scheme, advice and examples were given to examiners to make sure that candidates who were able to successfully navigate this logic chain were credited.  **Misconception** Checking whether a room is either basic or premium can be done in multiple ways. Candidates can either check for the positive (i.e. check that it is either basic or premium) or check that for the negative (i.e. check whether it is something else). However, there are many common errors that were seen : • `IF room == “basic” or “premium”` is **incorrect** as the second part of the statement is not evaluated against anything. This was perhaps the most common mistake. • `IF room == “basic” or room == “premium”` is **correct** and checks for validity. • `IF room == basic or room == premium` is **incorrect** as the lack of string delimiters means that basic and room would be treated as variables rather than strings. • `IF room != “basic” or room != “premium”` is also **incorrect**. This checks for invalid input but because **or** is used, only one condition needs to be True for the whole statement to be True. This means that if **basic** is entered, it would be classed as invalid (as it isn’t premium) and vice-

Question	Answer/Indicative content	Marks	Guidance

			versa. There is no way for any entry in this example to be classes as valid.
			• `IF room != "basic" and room != "premium"` is **correct**. This checks for invalid inputs but needs both conditions to be True.
			
			The same explanation follows for the other two necessary checks.
			
			Exemplar 3
			
			 
			`if firstName == "" OR surname == "" then`
			`print("NOT ALLOWED")`
			`else`
			`if room != "basic" AND room != "premium" then`
			`print("NOT ALLOWED")`
			`else`
			`if nights < 1 OR nights > 5 then`
			`print("NOT ALLOWED")`
			`else`
			`print("ALLOWED")`
			`endif`
			`endif`
			`endif`
			
			This exemplar shows a fully correct response. The candidate checks for invalid responses and correctly uses Boolean operators to check multiple criteria at each step. If any check returns True, "Not allowed" is printed and the program ends. Efficient use of `if … else …` means that the next check only proceeds if the previous check returns False.
			
			If all three checks return False, the final else is triggered to print "Allowed".
			
			It must be noted that this is only one way of achieving full marks. An equivalent program that checks for valid responses at each turn would also be possible. Candidates should be encouraged to use whatever structure they feel is sensible. If a response can logically be followed then it will achieve high marks.

<table>
<tr><th colspan="3">Question</th><th>Answer/Indicative content</th><th>Marks</th><th>Guidance</th></tr>
<tr><td></td><td></td><td>ii</td><td>• Normal
• 1 or 5 (not 0 or 6 as says allowed)
• Any numeric value except 1 to 5 // any non-numeric input (e.g. "bananas")</td><td>3
(AO3 2c)</td><td>Allow other descriptions that mean normal (e.g. valid / typical / acceptable)

<table><tr><th>Test data (number of nights)</th><th>Type of test</th><th>Expected output</th></tr><tr><td>2</td><td>Normal</td><td>ALLOWED</td></tr><tr><td>1 // 5</td><td>Boundary</td><td>ALLOWED</td></tr><tr><td>e.g. 7</td><td>Erroneous/Invalid</td><td>NOT ALLOWED</td></tr></table>
<u>Examiner's Comments</u>

This question was answered well by the majority of candidates.</td></tr>
<tr><td></td><td>c</td><td>i</td><td>• Function header for newPrice…
• …taking (at least) two parameters
• …correctly calculates price based on parameters (if present) <u>within function</u> …/
• … returns this calculated price</td><td>4
(AO3 2b)</td><td>BP1 must be clear that a new function is being defined. E.g. <code>function / def</code> keyword. Allow FT for subsequent marks if not present.

Ignore any code outside attempt at function definition.

Ignore additional parameters. Ignore inputs or additional code as long as these do not overwrite parameters or affect operation of function.

If inputs used instead of parameters, FT for BP3. Allow use of <code>else</code> for second room type in BP3.

Attempt at calculation needed to award BP4. Must return (not output) value. Return can be done e.g. in VB by assigning to function name (e.g. <code>newPrice = price</code>)

e.g.

<pre>function newPrice(nights, room)

  if room == “basic” then

  if room== 60 * nights then

  elseif room == “premium” then

  price = 80 * nights

  endif</pre></td></tr>
</table>

Question	Answer/Indicative content	Marks	Guidance

			`return price`
			
			`endfunction`
			
			Examiner's Comments
			
			Defining functions appeared to be a concept that candidates did not fully understand.
			
			Where a candidate did not attempt to define a function and instead simply calculated the price needed, very few (if any) marks were available.
			
			Successful responses could have been constructed from any suitable function definition keyword such as `function` (OCR ERL, VB, JavaScript, etc), `def` (Python) or others. Answers in C#, Java or other languages referring to methods were also accepted.

Question			Answer/Indicative content	Marks	Guidance

		ii	• Call function newPrice…
			• …with `("premium", 5)` as parameters
			• … Output **returned value**
				3 (AO3 2b)	Order of parameters not important
					
					"premium" must use string delimiters (e.g. "quotes")
					
					e.g.
					
					`print(newPrice("premium", 5))`
					
					`x = newPrice(5, "premium")`
					
					`print(x)`
					
					Do not allow function definitions for BP1
					
					Ignore capitalisation of `newPrice`
					
					Candidate could store returned value in a variable and then print this, or store parameters in variables before passing in - these are all acceptable
					
					Ignore any superfluous code given
					
					Do not credit answers where `newPrice` is overwritten prior to use.
					
					Ignore spaces. Allow function call if brackets missing (e.g. `newprice` instead of `newprice()`)
					
					Examiner's Comments
					
					Even if candidates were not able to create a function, this question was independent to (i) and so marks were available for simply using the function to output a value.
					
					Candidate found this question challenging. Many candidates called the function but most did not understand that the room type was a string and so required string delimiters (e.g. quotation marks) around the parameter.
					
					Where candidates defined local variables, assigned the values needed to the variables and then passed these into the function call were accepted.

Question	Answer/Indicative content	Marks	Guidance

d	• Inputs hours AND electric (two separate inputs), storing or using these.
	• Checks if car is electric (IF/Select statement)…
	• … correctly calculates and outputs price (hours * 2 // price / 2) for electric
	• … correctly calculates and outputs price (hours * 4 // electric price * 2) for non-electric
	• Attempt at repetition of BP1 to 4…
	• …until 0 hours entered
		6
		(AO3 2c)
			Initialisation of price and hours not necessary, but if present hours must be non-zero for BP6 to be given.
			
			BP5 must include all points attempted. Can still be credited if any of BP1 to 4 not attempted / incorrect.
			
			BP6 can be given as FT even if BP5 (loop) is in the wrong place / does not include all required code
			
			BP6 could be achieved as repeated function calls / recursion
			
			Initial input outside of loop that is then <u>also</u> included within loop is fine. For example, input of hours outside of loop but input is then repeated again at end of loop.
			
			Do not accept `while hours > 0` (could be -1)
			
			Do not penalise answers where 0 is output when loop exits
			
			e.g.
			
			`while hours != 0`
			`  hours = input(”Enter hours”)`
			`  electric = input(“enter Y for electric or N”)`
			`  if electric == ”Y” then`
			`  price = hours * 2`
			`  elseif electric == ”N” then`
			`  price = hours * 4`
			`endif`
			`print(price)`
			`endwhile`
			
			<u>Examiner’s Comments</u>
			
			This question was relatively well answered by candidates.
			
			Candidates were generally able to create suitable high-level program code to calculate and output the total price based on the information given.
			
			Many candidates ignored the requirement to repeat until 0 was entered; in this case, 4 out of the 6 marks were still available.

Question			Answer/Indicative content	Marks	Guidance
					To achieve marks for iteration it needed to both repeat the correct parts of the program and correctly terminate as per the requirements given. A typical mistake was to repeatedly calculate the price but not ask afresh for new inputs.
			Total	**28**	
6			• SELECT **StudentName, Subject, Grade** • FROM **Results** • WHERE **Subject = "Art"**	1 (AO1 1b) 2 (AO3 2a)	Correct Answer Only Accept SELECT *
			Total	**3**	
7			• `freeseats` called with `"Red"` • …returned value assigned to variable `redseats`	2	`redseats = freeseats("Red")` "`Red`" must use suitable string delimiters (e.g. speech marks) if directly passing the string. Do not penalise case.
			Total	**2**	

Question			Answer/Indicative content	Marks	Guidance
8	a		One mark per correct choice • SELECT ItemCode, ItemName • FROM tblStock • WHERE Price >=60	3	Accept other markings that indicate a choice has been made (e.g. a cross, etc)

	b	i	One mark per bullet point, in the correct place
			
			• `size // len(discountcodes-1)`
			• `code`
			• `price // newprice`
			• `[x,1] // [x][1]`
			• `return newprice // checkdiscount = newprice`
				5	e.g.
					
					`function checkdiscount(price, code)`
					`  newprice = price`
					`  size = len(discount)-1`
					`  for x = 0 to` **`size`**
					`  if discount[x,0] ==` **`code`** `then`
					`  newprice =` **`price`** `– discount[`**`x,1`**`]`
					`  endif`
					`  next`
					`  return newprice`
					`endfunction`

		ii	One mark per bullet point, maximum 2 marks • `newprice` • `size` • `x`	2	Do not penalise capitalisation Accept price, code, discount

Question			Answer/Indicative content	Marks	Guidance

		iii	• asks for price **and** discount code to be input
			• …passes both to the checkdiscount() function as parameters…
			• ...stores / uses returned value
			• calculates total of all prices entered/returned
			• repeats until 0 is entered as price
			• outputs calculated total
				6	High-level programming language / OCR Exam Reference Language response required
					
					Do not accept pseudocode / natural language.
					
					BP3 allow total of prices entered as FT if candidate does not achieve BP2
					
					e.g.
					
					`total = 0`
					`do`
					`  price = input("Enter a price")`
					`  code = input("Enter a discount code")`
					`  newprice = checkdiscount(price, code)`
					`  total = total + newprice`
					`until price == 0`
					`print(total)`
					
					alternative example
					
					`total = 0`
					`price = 1`
					`while price != 0`
					`  price = input("Enter a price")`
					`  code = input("Enter a discount code")`
					`  total = total + checkdiscount(price, code)`
					`endwhile`
					`print(total)`

			Total	16	

Question			Answer/Indicative content	Marks	Guidance
9			`print (minsPlayed[0,4])`	1 (AO3 2b)	<u>High-level programming language / OCR Exam Reference Language response required</u> Do not accept pseudocode / natural English. `print` may be a suitable output command word that could be found in a HLL e.g. `print` (Python), `console.writeline` (VB), `cout` (C++) The array elements may be accessed together `[0,4]` (VB.NET) or separately `[0][4]` (Python)
			Total	1	