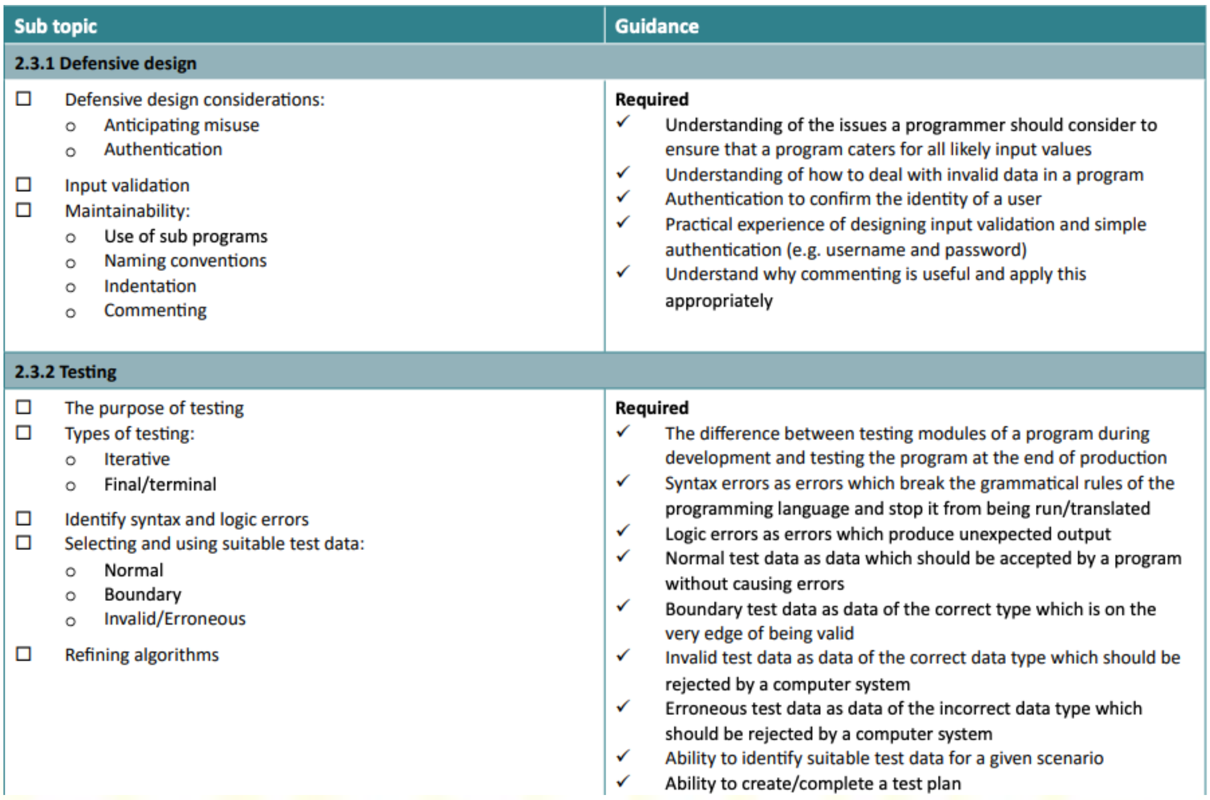
2.3 – Producing Robust Programs – Past Exam Questions



Sub topic	Guidance
2.3.1 Defensive design	
☐ Defensive design considerations: ○ Anticipating misuse ○ Authentication ☐ Input validation ☐ Maintainability: ○ Use of sub programs ○ Naming conventions ○ Indentation ○ Commenting	**Required** ✓ Understanding of the issues a programmer should consider to ensure that a program caters for all likely input values ✓ Understanding of how to deal with invalid data in a program ✓ Authentication to confirm the identity of a user ✓ Practical experience of designing input validation and simple authentication (e.g. username and password) ✓ Understand why commenting is useful and apply this appropriately
2.3.2 Testing	
☐ The purpose of testing ☐ Types of testing: ○ Iterative ○ Final/terminal ☐ Identify syntax and logic errors ☐ Selecting and using suitable test data: ○ Normal ○ Boundary ○ Invalid/Erroneous ☐ Refining algorithms	**Required** ✓ The difference between testing modules of a program during development and testing the program at the end of production ✓ Syntax errors as errors which break the grammatical rules of the programming language and stop it from being run/translated ✓ Logic errors as errors which produce unexpected output ✓ Normal test data as data which should be accepted by a program without causing errors ✓ Boundary test data as data of the correct type which is on the very edge of being valid ✓ Invalid test data as data of the correct data type which should be rejected by a computer system ✓ Erroneous test data as data of the incorrect data type which should be rejected by a computer system ✓ Ability to identify suitable test data for a given scenario ✓ Ability to create/complete a test plan

2022

4 Jack is writing a program to add up some numbers. His first attempt at the program is shown.

```
a = input("Enter a number")
b = input("Enter a number")
c = input("Enter a number")
d = input("Enter a number")
e = input("Enter a number")
f = (a + b + c + d + e)
print(f)
```

(a) Give **two** ways that the maintainability of this program could be improved.

1 ....

....

2 ....

....

[2]

(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 marks

(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]

Sample Paper

7 The area of a circle is calculated using the formula π × r² where π is equal to 3.142 and r is the radius.

A program is written to allow a user to enter the radius of a circle as a whole number between 1 and 30, then calculate and output the area of the circle.

```
01  radius = 0
02  area = 0.0
03  radius = input("Enter radius")
04  if radius < 1 OR radius > 30 then
05  print("Sorry, that radius is invalid")
06  else
07  area = 3.142 * (radius ^ 2)
08  print (area)
09  endif
```

(a) Explain, using examples from the program, **two** ways to improve the maintainability of the program.

1 ....

....

....

....

2 ....

....

.... **[4]**

(b) The program should only allow values from **0** to **300** inclusive as valid inputs. If the data entered breaks this validation rule, an error message is displayed.

(i) Complete the following program to output `"Invalid input"` if the data does not meet the validation rule.

You must use **either:**

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

```
mins = input("Enter minutes played: ")
if mins < 0 ....................... mins ....................... then
     ................................ ("Invalid input")
endif
```

[3]

(ii) Complete the following test plan for the program in **8(b)(i)**.

Test data	Test type	Expected result
25	Normal	Value accepted
	Invalid	`Invalid input` message displayed
300	Boundary	

[2]

(f) A teacher writes an algorithm to store the name of the game a student plays each night (for example "OCR Zoo Simulator").

`variable.length` returns the number of characters in `variable`.
`variable.upper` returns the characters in `variable` in upper case.

```
valid = false

while(valid == false)

    gameName = input("Enter the game name")

    if (gameName.length > 0) AND (gameName.length < 20)

        gamesPlayed = gameName.upper

        valid = true

        print("Valid game name")

    else

        print("Game name is not valid")

    endif

endwhile
```

The algorithm needs testing to make sure the IF-ELSE statement works correctly.

Identify **two** different pieces of test data that can be used to test different outputs of the algorithm. Give the output from the program for each piece of test data.

Test data 1 ....

Expected output ....

Test data 2 ....

Expected output ....

[4]

2021

6 OCRBlocks is a game played on a 5 × 5 grid. Players take it in turns to place blocks on the board. The board is stored as a two-dimensional (2D) array with the identifier `gamegrid`

Fig. 6.1 shows that players A and B have placed three blocks each so far.

	0	1	2	3	4
0	A			B	
1					
2		B			
3	A		B		
4			A		

Fig. 6.1

The function `checkblock()` checks whether a square on the board has been filled. When `checkblock(4,2)` is called, the value "A" is returned.

```
function checkblock(r,c)
    if gamegrid[r,c] == "A" or gamegrid[r,c] == "B" then
       outcome = gamegrid[r,c]
    else
       outcome = "FREE"
    endif
    return outcome
endfunction
```

(c) When `checkblock(-1,6)` is called, an error is produced.

(i) State why this function call will produce an error.

....

.... **[1]**

(ii) Describe how validation could be added in to the `checkblock()` function to stop this error from occurring.

....

....

....

....

....

.... **[3]**

2020

3 A vending machine has the following options available.

Item code	Item name	Price
A1	Crisps, bacon flavour	£0.75
A2	Crisps, salted	£0.75
B1	Chocolate bar	£0.90
C1	Apple pieces	£0.50
C2	Raisins	£0.85

Users insert coins into the vending machine and then enter the two character item code of their selection. If the user has inserted enough money, the vending machine will release the chosen item and output any change required. If the user enters an invalid item code then a suitable error message is displayed.

(a) The vending machine is tested before it is released.

(i) Explain the purpose of testing the vending machine.

....

....

....

.... **[2]**

(ii) Describe the difference between iterative testing and final testing.

....

....

....

.... **[2]**

(iii) Complete the following test plan for the vending machine.

Code entered	Money inserted	Expected result
B1	£1	Chocolate bar served, £0.10 change given
	£0.85	Raisins served, no change given
C1		Error – not enough money inserted
C3	£0.75	

[3]

(b) The algorithm for one section of the vending machine program is shown in pseudocode.

```
if money >= price then
    venditem()
    giveChange(money - price)
else
    print("Error - not enough money inserted")
endif
```

(d) When writing the program for the vending machine, maintainability was considered.

(i) Identify **two** ways that the program in **part (b)** has been made more maintainable.

1 ....

....

2 ....

.... **[2]**

(ii) Give **one** additional way that the maintainability of the program can be improved.

....

.... **[1]**

2019

4 Elliott plays football for OCR FC. He wants to create a program to store the results of each football match they play and the names of the goal scorers. Elliott wants individual players from the team to be able to submit this information.

(b) Describe **two** examples of defensive design that should be considered when developing this program.

1 ....

....

....

....

....

....

2 ....

....

....

....

....

.... **[4]**

2017

(e) The program is being extended to ask the user to enter numbers into the array. An algorithm is written to check that the input is valid.

```
do
    input Number
until Number >= 0 AND Number <= 100
```

State **one** item of borderline data and **one** item of invalid data that can be input to test the algorithm works correctly.

Borderline ....

Invalid ....

[2]