2.3 – Producing Robust Programs – Past Exam Questions – Solutions

2022

4	**(a)**		Any two bullet points for one mark each: • Add comments • Name variables sensibly • **Put into** subroutine / procedure / function • Use loop / iteration	2 **(AO2 1b)**	Do not accept indentation (no code to sensibly indent in this example) "Use a subroutine" is not enough. Must be clear that existing code will be put into a new subroutine.

5	**(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 any of the 3 checks are invalid (must check all three)
			• …Outputs "ALLOWED" (or equivalent) only 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 all three checks 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.
					
					Do not accept logically incorrect Boolean conditions such as
					`if firstname or surname == ""`
					
					Do **not** accept ≥ or ≤ for >=, <=. Ignore capitalisation
					
					e.g.
					`valid = True`
					`if firstname == "" or surname == "" then`
					`    valid = False`
					`end if`
					`if room != "basic" and room != "premium" then`
					`    valid = False`
					`endif`
					`if nights < 1 or 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 != "" and surname != ""`
					• `if room == "basic" or room == "premium"`
					• `if nights >= 1 and nights <= 5`

5	**(b)**	**(ii)**	• 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")	3 **(AO3 2c)**	Allow other descriptions that mean normal (e.g. valid / typical / acceptable)

Test data (number of nights)	Type of test	Expected output
2	Normal	ALLOWED
1 // 5	Boundary	ALLOWED
e.g. 7	Erroneous/Invalid	NOT ALLOWED

Sample Paper

7	a		1 mark for naming the example and 1 mark for an example related to that method E.g • Comments/annotation… • …E.g. any relevant example, such as line 4 checks the input is valid • Indentation… • …E.g. indenting within IF statement • Using constants… • …E.g. π	**4** **(AO2 1b)**	
8	b	i	• `or` • `>300 // >= 301` • `print`	**3** **(AO3 2b)**	**High-level programming language / OCR Exam Reference Language response required** Do not accept pseudocode / natural English. MP2 do not accept 'greater than', must use the HLL syntax > or >= MP3 must be a suitable output command word that could be found in a HLL e.g. `print` (Python), `console.writeline` (VB), `cout` (C++)
	b	ii	• Suitable invalid test data (i.e. > 300, e.g. 350) • "Value accepted" or equivalent	**2** **(AO3 2c)**	
8	f		1 mark per bullet • Test data either 0 or less characters, or 20 or more characters • Stating correct output • Test data between 1 and 19 characters (inc) • Stating correct output	**4** **(AO3 2c)**	Mark test data first, both must meet different criteria. Then mark output for each.

2021

	(c)	(i)	• Parameter values outside index range / larger than 4 / smaller than 0 // -1, 16 is not a valid block	1	Answer must refer to either array or gameboard / grid / block
		(ii)	• Use selection / IF / Switch-Case / range check • …check that **parameters** are >=0 **and** <= 4… • **…Return** error code if invalid // set outcome to error	3	Allow equivalent checks (e.g. <5, between 0 and 4) for BP2 Allow reference to `r` and `c` as parameters. BOD handle error for BP3 (e.g. repeat until valid) Answer must be a description, code by itself is NAQ

2020

3	(a)	(i)	1 mark per bullet to max 2
			e.g.
			• Check the program meets the **user requirements**
			• Check the program works (as intended) // detect logic / syntax errors
			• Check the program does not crash (under invalid entry) // check error messages are suitable
			• …allow these errors to be fixed
			• …make sure there are no problems when released
			• Any suitable example **related to the vending machine** e.g. gives correct change
				2
				
				AO1 1b(2)
					Allow two any suitable examples for two marks
					
					BOD "find errors", "find bugs" for BP2
					
					"fix errors" by itself is one mark (BP4)

3	(a)	(ii)	1 mark per bullet to max 2 • Iterative is during development // repeatedly testing after/while making changes • Final is when the development is (almost) complete // done after iterative testing	2 AO1 1b(2)	Do not accept just "repeatedly testing" for iterative BOD "iterative testing tests modules/sections"
3	(a)	(iii)		3 AO3 2b(3)	For £0.49 accept any value <£0.50. Must be a specific value, not a description. Accept any suitable error message for invalid selection

Code entered	Money inserted	Expected result
C2		
	£0.49 **(or any value less that £0.50)**	
		Invalid Selection **(or any suitable error message)**

3	**d**	**i**	1 mark per bullet to max 2 • Indentation // whitespace • Appropriately named variables / identifiers • Modularisation / use of subroutines	2 AO2 1b(2)	
3	**d**	**ii**	• Comments • Use of constants	1 AO2 1b(1)	

2019

4	**(b)**		1 mark per bullet, mark in pairs. Max 2 per point. e.g. • Input sanitisation • …cleaning up input data / removing unwanted data • …by example (e.g. removing special characters / preventing SQL injection) • Validation • …checking whether input data should be allowed / is sensible / follows criteria • …by example (e.g. goals cannot be less than 0) • Verification • … checking whether data has been entered correctly • …by example (e.g. double entry / visual check) • Authentication • …ensuring only allowed / authorised users can gain access • …by example (e.g. usernames /passwords) • Maintainable code • …to allow other programmers to understand the code • …by example(e.g. comments, indentation, meaningful variable names)	4 AO2 1a (2) AO2 1b (2)	Mark first answer only in each section For validation, allow one example of a type of validation (e.g. type check, range check) e.g. question so allow other sensible examples such as audit logging, encryption of data Do not allow "data is correct" as expansion for validation – validation checks that data is sensible or follows rules, NOT that it is correct. Planning for contingencies and anticipating misuse are not examples by themselves, but discussion of these may fit under other points – e.g. input sanitisation, validation.

2017

3	**e**		1 mark for sensible borderline data, 1 mark for sensible invalid data. • Borderline – 0, 100 • Invalid – number less than 0 (eg -1, -12) / number more than 100 (eg 101, 206) / non-numeric data (eg "test", "#!*%")	**2**	