GCE

Computing

Advanced Subsidiary GCE

Unit **F452:** Programming Techniques and Logical Methods

Mark Scheme for January 2012

Oxford Cambridge and RSA Examinations

This mark scheme is published as an aid to teachers and students, to indicate the requirements of the examination. It shows the basis on which marks were awarded by examiners. It does not indicate the details of the discussions which took place at an examiners' meeting before marking commenced.

All examiners are instructed that alternative correct answers and unexpected approaches in candidates' scripts must be given marks that fairly reflect the relevant knowledge and skills demonstrated.

Mark schemes should be read in conjunction with the published question papers and the report on the examination.

OCR will not enter into any discussion or correspondence in connection with this mark scheme.

Any enquiries about publications should be addressed to:

OCR Publications
PO Box 5050
Annesley
NOTTINGHAM
NG15 0DL

Telephone: 0870 770 6622
Facsimile: 01223 552610
E-mail: publications@ocr.org.uk

Annotations

Annotation	Meaning
^	Omission mark
BOD	Benefit of doubt
C	Subordinate clause/Consequential error
Cross	Cross
E	Expansion of a point
FT	Follow through
NAQ	Not answered question
NBOD	Benefit of doubt not given
P	Point being made
REP	Repeat
/	Slash
Tick	Tick
TV	Too vague
ZERO	Zero (big)

Question			Answer	Marks	Guidance
1	(a)	(i)	• The program is nearly finished/pre-released version • Testing by third party users/not the programmers • … eg a focus group of library members/other relevant example • Tested in normal use/as intended • They report (any errors they find) back to Marek / Marek addresses errors found	3	
		(ii)	• The programme is completed • all errors, which have been found, have been solved • Marek demonstrates the program to the library staff/library staff use program • Purpose is to show that it meets the agreed requirements/fit for purpose • Programme is used with live data	3	
	(b)		For each test: • 1 mark for test data (provided it matches the type of test data) • 1 mark for expected outcome if it matches test data Normal • Any valid date not before issue date (fine or no fine message) Borderline • 1st February (no fine message) • 14th February (no fine message) • 15th February (fine/no fine message) • 16th February (fine message) Invalid • Return date before 01/02/2012 (error message) • Date does not exist in the month (eg 30/02) (error message)	6	A date must be given as the test data. If a description is given (e.g. after 5 days) mark incorrect but award a BOD for a correct outcome, PROVIDED the description is not “within 14 days” or “after 14 days”.

Question			Answer	Marks	Guidance
			• Date not entered correctly (error message)		
	(c)		• DateBorrowed: Date (Date/time)/Integer • DaysBorrowed: Integer • Fine: Real/Currency/Double/Float • BankCardNumber: String/Alphanumeric/Text • TransactionOK: **Boolean**	5	Accept other known equivalent
	(d)		• Speech synthesis… • eg to give the members instructions • Buzz/Beep/click etc (not "sound") • … reinforce (on screen) feedback / eg to signify transaction completed successfully/error • Alarm… • … to alert library staff / request for assistance	4	Second mark not dependent on first
2	(a)	(i)	Serial file • Records are stored in the order in which they are entered • Each new record is appended to the existing records • To find an item, you have to search from the first record until you get to the item you want Random file • The address of each record is calculated from the key.... • using a hash algorithm • Records may not be contiguous	4	Max 2 marks per file type
		(ii)	EITHER • The sequence of the data is relevant • … to show progress	2	

Question			Answer	Marks	Guidance
			OR • The data are likely to be all used (in sequence) • … eg to draw graph/calculate average		
	(b)		• < or <= • NextDate • DataFile	3	In this question, ignore case.
	(c)		• To initialise the value (of the variable/FastestTime)... • before it is used in a comparison operation (in line 07)/Unless a faster time is found, it will take the first time as the fastest time	2	
	(d)	(i)	• Executing lines of code repeatedly • Line 6 to 10 or 5 to 11 (repeated for each record in the file)	2	Accept using a loop **Accept line 5**
		(ii)	• Use a condition to decide if code should be executed • Lines 8 and 9 only executed if the condition on line 7 is true	2	**Accept line 7 (to 10)**
	(e)		Example: `OUTPUT "The fastest time is” + FastestTime + “on” + DateOfFastestTime` Award marks for: • Concatenation used… • … to make a user friendly sentence.	2	Any concatenation operators/variables mistakenly put in a literal string (ie in quotes) should be treated as part of the string

Question			Answer	Marks	Guidance

	(f)		Example
			`OPEN DataFile (FOR INPUT/READ)`
			`WHILE DataFile is not at end of file`
			`   READ NextDate, NextTime FROM DataFile`
			`END WHILE`
			`OUTPUT NextTime`
			`CLOSE DataFIle`
			
			Award marks for:
			• Opening the file correctly (for input/read)
			• Reads at least one line
			• Loop with correct condition
			• Output the time in the last item
			• Close/release the file
				5	In the last two bullet points order can be reversed

3	(a)		• To determine/force the order in which the operations are carried out • To make the expression easier to understand • + a relevant example from the expression	2	If the example fully explains the parentheses in the question award full marks.
	(b)		• (c+d) >= 180 is TRUE • (a+b+c+d) > 320 is FALSE • TRUE AND FALSE = FALSE	3	
	(c)	(i)	• A (symbolic) name given to a variable/subroutine etc…	1	
		(ii)	• a,b,c,d • HasDoneIt	2	Must be in correct case
		(iii)	• Give the identifiers meaningful names • … instead of a, b, c , d some examples of suitable names are given	2	

Question			Answer	Marks	Guidance
	(d)		eg • Indentation/show control structures with the lines within them indented • … so you can see where each structure starts and ends • Lines 2-6 indented • Comments/text/annotation which is ignored by the translator • … to explain the algorithm further • Example of a suitable comment • Avoid complex expressions like line 06 • And break them down step by step (so it is clearer whether they are correct) • … example code given Up to 3 marks each. Must give an example to get the third mark.	6	Accept Lines 2-5 being indented
4	(a)		• Bank note: £20 • Coins:£2, £2 and £1	2	
	(b)	(i)	Example: 2 * Count_of_2s + Count_of_1s • 2 * Count_of_2s • + Count_of_1s (* 1)	2	
		(ii)	Example: Total_value_of_coins = 2 * Count_of_2s + Count_of_1s • Recognisable assignment operator (=, :=, ←) • Direction of assignment is correct.	2	FT from part (i) is implicit. It is the assignment that is being assessed here.
		(iii)	Example: Total_value_of_coins < 20 • Uses Total_value_of_coins • Correct inequality	2	

Question			Answer	Marks	Guidance
	(c)		• A subroutine/named section of code (to perform a task) • Can be called from parent program/returns control to parent program when complete • Used as a statement in the main program	3	**Accept:** • Does not return a value / Can change values **from the main program** passed by reference
	(d)	(i)	• It does not dispense £2 coins when there is £2 credit • Should be If Credit_left >= 2 • Type of Error: Logic Error or • Undeclared/wrong identifier • Should be Credit.left instead of Credit_left • Type of Error: Syntax/Logic Error	3	
		(ii)	• There is a missing END IF • There if are two IF statements • Type of Error: Syntax error	3	Award 2nd bullet if candidate identifies an IF without the end if on line thre
		(iii)	• Any other error not mentioned	1	Accept Arithmetic error
	(e)		• Breakpoints will cause the code to stop at specified lines of code • Stepping allows program to be executed a line at a time • Allow programmer to check the values of variables • Allow programmer to see the flow of control • eg to determine logic error / check results of calculations • eg to determine at what point an error occurs • eg to detect undeclared identifiers • eg to determine why a run time error occurred	4	

Question			Answer	Marks	Guidance	
					Content	Levels of response

4	(f)		Example:
			
			`PROCEDURE Button_Auto_Pressed`
			
			`{Give as many £2 coins`
			`as possible}`
			`WHILE Credit_left >= 2 and Count_Of_2s > 0`
			`Drop_2_pound_coin`
			`Count_Of_2s = Count_of_2s – 1`
			`Credit_left = Credit_left – 2`
			`END`
			
			`{Give the rest in £1 coins}`
			`WHILE`
			`WHILE Credit_left > 0`
			`Drop_1_pound_coin`
			`Count_Of_1s = Count_Of_1s – 1`
			`Credit_left = Credit_left – 1`
			`END WHILE`
			
			`END PROCEDURE`
				8	The use of meaningful variable names should be considered when judging how well the algorithm is annotated.
					
					The algorithm will typically include:
					• Dispenses £2 coins …
					• … until credit is < £2
					• … or £2 coins run out
					• If credit > 0
					• … Dispenses £1 coins
					• … until credit is 0
					• Updates Count_of_2s correctly
					• Updates count_of_1s correctly
						High level response (6–8 marks)
						Candidate offers a complete, working algorithm which both shows clearly how the variables are updated as well as uses a clear strategy to determine how to give the change using the coins available (eg use as many 2s as possible, then 1s). The algorithm is in a well annotated and correctly structured format eg pseudocode with indentations, correctly numbered statements, or a well organised flowchart. Technical terms and spelling will be used appropriately and correctly.
						Medium level response (3–5 marks)
						Candidate has an algorithm which is not fully explained or contains some errors either in the update of the variables, or in the strategy for determining what change to give. There is an attempt to structure the code correctly but may contain some errors, however the overall structure of the code can still be understood. Technical terms and spelling are mostly correct.
						Low level response (0–2 marks)
						Candidate's algorithm neither shows fully how the variables are updated nor what the strategy is for determining the change. The code is poorly structured and not structured at all, and errors with spelling and technical terms make the algorithm difficult to understand.

Question			Answer	Marks	Guidance
5	(a)		• Easier to write a module at a time • Easier to test individual modules • Easier to understand/debug a single module (eg by a third party) • Modules from other programs can be reused • Modules can be written by other programmers • … according to their expertise • Modules can be replaced without affecting whole program • Modules can be written in different languages (as appropriate)	3	Award other relevant points if well explained
	(b)		• SELECT CASE allows you to branch on multiple values (of the same variable)/easier to add more options • IF statements give 2 options at a time/many (nested) IF statements would be needed	2	Must be a clear comparison of the SELECT and IF statements to get both marks.
	(c)	(i)	• Line 16	1	
		(ii)	• Remove line 16 • Use a loop containing lines 2 to 15 • … and which ends when the command Quit is entered	3	Accept answers where the candidate has attempted code which addresses the bullet points given. Award a mark for an infinite loop if the candidate clearly indicates that the loop will exit early if the command Quit is entered.
		(iii)	• Saves memory/will not run out of stack space • … as all previous calls to the menu are not stored	2	

OCR (Oxford Cambridge and RSA Examinations)
1 Hills Road
Cambridge
CB1 2EU

OCR Customer Contact Centre

Education and Learning
Telephone: 01223 553998
Facsimile: 01223 552627
Email: general.qualifications@ocr.org.uk

www.ocr.org.uk

For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored

Oxford Cambridge and RSA Examinations
is a Company Limited by Guarantee
Registered in England
Registered Office; 1 Hills Road, Cambridge, CB1 2EU
Registered Company Number: 3484466
OCR is an exempt Charity

OCR (Oxford Cambridge and RSA Examinations)
Head office
Telephone: 01223 552552
Facsimile: 01223 552553