Level 3 Technical Level
IT: PROGRAMMING
F/507/6465

Unit 2 Computer programming

Mark scheme

January 2019

Version: 1.0 Final

1 9 1 A F 5 0 7 6 4 6 5 / M S

Mark schemes are prepared by the Lead Assessment Writer and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation events which all associates participate in and is the scheme which was used by them in this examination. The standardisation process ensures that the mark scheme covers the students' responses to questions and that every associate understands and applies it in the same correct way. As preparation for standardisation each associate analyses a number of students' scripts. Alternative answers not already covered by the mark scheme are discussed and legislated for. If, after the standardisation process, associates encounter unusual answers which have not been raised they are required to refer these to the Lead Assessment Writer.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of students' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Further copies of this mark scheme are available from aqa.org.uk

Level of response marking instructions

Level of response mark schemes are broken down into levels, each of which has a descriptor. The descriptor for the level shows the average performance for the level. There are marks in each level.

Before you apply the mark scheme to a student's answer read through the answer and annotate it (as instructed) to show the qualities that are being looked for. You can then apply the mark scheme.

Step 1 Determine a level

Start at the lowest level of the mark scheme and use it as a ladder to see whether the answer meets the descriptor for that level. The descriptor for the level indicates the different qualities that might be seen in the student's answer for that level. If it meets the lowest level then go to the next one and decide if it meets this level, and so on, until you have a match between the level descriptor and the answer. With practice and familiarity you will find that for better answers you will be able to quickly skip through the lower levels of the mark scheme.

When assigning a level you should look at the overall quality of the answer and not look to pick holes in small and specific parts of the answer where the student has not performed quite as well as the rest. If the answer covers different aspects of different levels of the mark scheme you should use a best fit approach for defining the level and then use the variability of the response to help decide the mark within the level, ie if the response is predominantly level 3 with a small amount of level 4 material it would be placed in level 3 but be awarded a mark near the top of the level because of the level 4 content.

Step 2 Determine a mark

Once you have assigned a level you need to decide on the mark. The descriptors on how to allocate marks can help with this. The exemplar materials used during standardisation will help. There will be an answer in the standardising materials which will correspond with each level of the mark scheme. This answer will have been awarded a mark by the Lead Examiner. You can compare the student's answer with the example to determine if it is the same standard, better or worse than the example. You can then use this to allocate a mark for the answer based on the Lead Examiner's mark on the example.

You may well need to read back through the answer as you apply the mark scheme to clarify points and assure yourself that the level and the mark are appropriate.

Indicative content in the mark scheme is provided as a guide for examiners. It is not intended to be exhaustive and you must credit other valid points. Students do not have to cover all of the points mentioned in the Indicative content to reach the highest level of the mark scheme.

An answer which contains nothing of relevance to the question must be awarded no marks.

The following annotation is used in the mark scheme:

; - means a single mark

// - means alternative response

/ - means an alternative word or sub-phrase

A - means acceptable creditworthy answer

R - means reject answer as not creditworthy

NE - means not enough

I - means ignore

DPT - in some questions a specific error made by a candidate, if repeated, could result in the candidate failing to gain more than one mark. The DPT label indicates that this mistake should only result in a candidate losing one mark on the first occasion that the error is made. Provided that the answer remains understandable, subsequent marks should be awarded as if the error was not being repeated.

Question	Guidance	Marks
01 **Option A**	**Mark is for AO3** an array of characters; **R.** If more than one box ticked	**1**
02 **Option D**	**Mark is for AO1** a way to classify programming languages based on features; **R.** If more than one box ticked	**1**
03 **Option D**	**Mark is for AO1** a timer expiring; **R.** If more than one box ticked	**1**
04 **Option A**	**Mark is for AO3** catalogues vulnerabilities in software or firmware; **R.** If more than one box ticked	**1**
05 **Option D**	**Mark is for AO2** UML is a standard way to visualise the design of a system; **R.** If more than one box ticked	**1**

Question	Guidance	Marks
06.1	**Mark is for AO3** 20; A. any answer which shows understanding that the function returns 20, eg calcArea()=20	**1**
06.2	**Mark is for AO3** 45; **A.** x1y1=45	**1**
06.3	**2 marks for AO3** **1 mark** for each correct point, up to a maximum of 2 marks. **Examples include:** • A new variable (x1y1) is created (as a local variable) in the function; and this would not be returned // and this cannot be used outside the function; • The variable would be local to the function; so the output on line 5 would be 0; • The output on line 5 would be the variable defined in line 1 (ie zero); • x1y1 (on line 8) will now be 5*9 (45);	**2**
07.1	**Mark is for AO3** 1; **R.** 1 bit	**1**
07.2	**Mark is for AO3** (usually) 1; **A,** 2 (eg Java)	**1**

Question	Guidance	Marks
08	**6 marks for AO4** **1 mark** for each correct point or expansion point, up to a maximum of 6 marks. **Examples include:** • Modern computers have so much memory // processing capability that efficiency is less important; • It takes time to make code efficient // less time for compiler; • Efficient can mean ‘run faster’ or ‘code smaller’; • (Compilers are programmed by humans so) the programmer might find more efficient solutions themselves // humans make errors and don’t always notice things that a compiler might; • What makes a solution efficient depends on user/client requirements; the compiler might not notice structures / design requirements; • Users/organisations using old equipment/pressure on system resources; • Readability of code; • (If the code is at optimum efficiency) it is easier for other programmers to maintain; • It is good programming practice to make code as simple as possible; • A programmer may learn more about programming by learning how to make code efficient themselves; • Other developers/clients may understand the meaning/purpose of the code better if it is written efficiently; • Example of how to make code more efficient, eg abstraction, subroutines;	6

Question	Guidance	Marks
09	**3 marks for AO2** **1 mark** for each correct point or expansion point, up to a maximum of 3 marks. **Examples include:** • An informal/semiformal/intermediate/simple way of designing/describing a program/set of instructions; which can make the code/structures easier to understand; • Does not require any set programming language syntax; • Use of English/less technical language rather than programming terms; • Used to plan out / create a rough draft of a program; • Can be easily converted by someone who knows a particular programming language; but designed by a team with experience in differing languages (or none); • Summarises a program’s flow, but excludes underlying details // a rough sketch for a developer/client to get an idea of what to code;	3
10	**3 marks for AO3** **1 mark** for each type or subtype, up to a maximum of 3 marks. **Examples include:** • Semantic versioning; eg major/minor/patch/security fix; • Increment versioning; eg v1.0, build numbers; • Keyword versioning; eg o alpha; o beta; closed/private beta; open/public beta; o release candidate (RC); o release to manufacturing (RTM); o general availability (GA); o release to web (RTW); • Date versioning; eg date-time stamps; • Hash versioning // UUID versioning; eg hexadecimal string; • Tuple versioning; eg (1, 2, 3); **A.** 1 mark only for type and subtype, eg 1) date versioning, 2) date-time stamps	3

Question	Guidance	Marks
11	**3 marks for AO4** **1 mark** for each correct point or expansion point, up to a maximum of 3 marks. **Examples include:** • Software entities (classes, modules, functions, etc); should be open for extension but closed for modification**;** which means code which does not have to be modified every time the requirements change; • Should be able to extend a system without having to modify the system; • Parts of a program should be allowed to change the behaviour of another section without changing the behaviour itself; • Should be possible to add fields to the data structures of an open module; • Closed modules are available for use by other modules / parts of the system; if you modify these for a single purpose other parts will be affected; • A real-world example of usage, eg calculate the area of different shapes without modifying original function; **R.** Duplicate or similar points	3
12	**6 marks for AO4** **1 mark** for each factor, up to a maximum of 3 marks. **Examples include:** • Which device/platform it is for; • Screen pixels/DPI; • Screen dimensions/scale/proportion; • Input method; • Familiarity // consistency / simplicity of layout; • Responsiveness // ease of use / navigation / techniques eg use of carousels; • Colour scheme; • Compatibility with multiple browsers // applicable for the target audience; • File/download size // adverts; which may slow mobile phone browser down; • Accessibility; eg using the alt attribute; **1 mark** for each linked expansion point, up to maximum of 3 marks. **A.** Points which combine factors above, eg: • It is important to consider which device it is for; different devices have different screen dimensions // images might take longer to download; • It is important to consider the input methods used; so that they work with accessibility settings; **R.** Repeated/similar explanations/expansion points **R.** Screen resolution	6

Question	Guidance	Marks
13.1	**4 marks for AO2** **1 mark** for each correct point or expansion point, up to a maximum of 4 marks. **Examples include:** • Separates functionality into discrete modules; these can be put together to form the overall solution; the elements of one module will be detectable/can be referred to by other modules; • Decomposes a problem into a number of less complex sub-problems; • Independent modules with logically discrete functions; • Replace or add to any module without affecting the rest of system; • Multiple programmers divide up and debug the program independently; • Allows many programmers to collaborate/expertise to be split; • Ease of error identification / easier to maintain; • Optimise modules independently of other modules; • Interchangeable/re-use of modules in other programs; • Has rules, tools and methods that together prescribe how modules are deployed over time; • Closely related to structured programming and object-oriented programming; • Requires thorough documentation of modules; • Approach needs to be agreed across teams, eg naming conventions, versioning; • Reduces the need to repeat code;	**4**
13.2	**2 marks for AO2** **1 mark** for each advantage, up to a maximum of 2 marks. **Examples include:** • Reuse of code; • Scoping of variables easier; • Thorough testing; • Organisation of code, eg allows library programs to be inserted // programming codes shortened; • Errors localised // can be easily identified; • Develop/test functionality independently; • Simplified planning and engineering/standardisation; • More flexible/versatile use of functions; • Easier to maintain code // large programs easier to design and manage; • Different programmers can work on each module; shortening development time/more cost-effective; **R.** points already credited in 13.1.	**2**

Question	Guidance	Marks
14.1	**2 marks for AO3** **1 mark** for each correct point, up to a maximum of 2 marks. **Examples include:** • A marker which tells the debugger to stop execution // an intentional stopping point in a program; • To acquire knowledge about a program during its execution // a means of testing eg the programmer can inspect variables;	**2**
14.2	**4 marks for AO3** **1 mark** for naming each technique, up to a maximum of 2 marks. **1 mark** for a linked expansion point, up to a maximum of 2 marks. **Examples include:** • Watches; o enter variable names and expressions that you want to watch; o you use watch windows to monitor code; • Trace tables; o used to identify where (logical) errors occur in the program; o trace the value of variables (as each statement is executed); o help to visualise how an algorithm works; • Step into/mode; o executes code one statement at a time; o enters break mode at the current line of execution // or, in design mode, before; o steps into the next method call at the current line of execution; • Code inspection (manual/automated); o examining the computer source code to uncover errors and defects; o done manually, eg review team working with coders; o automated process (compiler/debugger) analyses for specific code quality issues as you edit; o warns of issues in the code/suggests fixes; **A.** white box / black box / closed beta **A.** error messages / syntax highlighting / read comments in the code **R.** similar points	**4**

Question	Guidance	Marks
15	**6 marks for AO1**	6

Mark using the levels of response table and the indicative content.

Descriptor	**Marks**
Clear understanding of the challenges of cross-platform development, eg: • Includes a range of indicative content with clear and relevant expansion points. • Discusses one or two areas in detail, perhaps with specific background knowledge applied.	5–6
Some understanding of cross-platform development and the challenges, eg: • A selection of indicative content with some relevant expansion points. • Discusses one area in detail, though the level of understanding is inconsistent.	3–4
Makes one or two points about cross-platform development, eg: • Makes one or two relevant points. • Attempts to discuss in detail, though some points either lack relevance, clarity or understanding.	1–2
No creditworthy material	0

Indicative content:

- Design specifications (eg Android v iOS v Windows).
- Speed (memory, processor, overall performance etc).
- Navigation controls/input methods/triggers (eg PC keyboard over joystick).
- Equivalent performance (eg gaming difficulty).
- Compatibility.
- Licensing.
- User base.
- Codebase / different languages.
- Technical expertise.
- Cost effectiveness of multiple versions // rapid change in technology.
- Operating system.
- Hardware platform, limitations (eg graphics capability, licensing).
- Compilers.
- Conventional vs mobile technology // different mobile operating systems.

Question	Guidance	Marks
16.1	**3 marks for AO3** **1 mark** for each relevant method, up to a maximum of 3 marks. **Examples include:** • Email validation, eg double entry, format checking, validation link sent to email address, check for @ symbol. • Checks against existing database (username, email address, etc). • Length check. • Presence check. • Format check. • Double entry. • Specific character combination. **R.** Repeated methods. **R.** Answers without a method or (if relevant) an expansion point, eg 'email validation'.	**3**

Question	Guidance	Marks
16.2	**12 marks for AO2** Mark the flowchart using the levels of response table below.	**12**

Descriptor	**Marks**
• Registration and login procedure both complete and logical • Sequences logical, loops work and iterate appropriately • Inputs and outputs provide a clear, authentic client/user experience • Decisions are broken down and paths labelled • Develops validation against individual inputs • Processes are developed, with clear labels/understanding • Flowchart symbols are appropriate and used consistently • Direction of flow accurate and marked clearly	9–12
• Registration and login procedure both logical • Most sequences logical, most loops work and iterate appropriately • Most inputs and outputs provide a valid client/user experience • Some decisions work and some paths labelled • Includes some validation • Some processes are shown, with some labels/understanding • Most flowchart symbols are appropriate • Direction of flow accurate	5–8
• Registration or login procedure present • Some logical sequences, loops attempted • Inputs and outputs attempted • Some decisions work • Attempts validation • A process is shown • Some flowcharts symbols are appropriate, perhaps inconsistent • Direction of flow is indicated in places	1–4
No creditworthy material	0

Question	Guidance	Marks
17.1	**12 marks for AO3** Award the following marks using the indicative content (trace table) on following page. **1 mark** for each appropriately named column, up to a maximum of 4 marks. **1 mark** for each correctly completed column, up to a maximum of 8 marks.	**12**

Indicative content:

Trace table for code in Figure 4

Line	win	draw	lose	**x loop counter**	**x==’W’**	**x==’D’**	**x<>’W’ AND x<>’D’**	Output
03	**0**	**?**	**?**	**?**				
04	**0**	**0**	**?**	**?**				
05	**0**	**0**	**0**	**?**				
07	**0**	**0**	**0**	**?**				**3 5 1**
10	**0**	**0**	**0**	**0**				
12	**0**	**0**	**0**	**0**	**TRUE**			
13	**1**	**0**	**0**	**0**				
14	**1**	**0**	**0**	**0**				
15	**1**	**0**	**0**	**0**				
16	**1**	**0**	**0**	**0**				
17	**1**	**0**	**0**	**0**				
10	**1**	**0**	**0**	**1**				
12	**1**	**0**	**0**	**1**	**FALSE**			
13	**1**	**0**	**0**	**1**				
14	**1**	**0**	**0**	**1**		**FALSE**		
15	**1**	**0**	**0**	**1**				
16	**1**	**0**	**0**	**1**			**TRUE**	
17	**1**	**0**	**1**	**1**				

A. different trace table conventions, eg may not repeat a value until it changes, True/False statements not repeated, blank instead of ‘?’.
A. x loop counter which goes from 1 to 2 (instead of 0 to 1).
A. other viable trace headings, eg ‘results’ variable.

Question	Guidance	Marks
17.2	**Mark is for AO3** **1 mark** for correctly completed table.	**1**

Line	win	draw	lose	Output
19	3	5	2	3 5 2

Question	Guidance	Marks
17.3	**2 marks for AO3** **1 mark** for each point, up to a maximum of 2 marks. **Examples include:** • An error in line 2 // results (list) includes ‘B’ when it should be ‘D’; • Line 17 counts anything that isn’t ‘W’ or ‘D’ as ‘L’; • Line 7 counts exact occurrences of ‘L’ (so ‘B’ is not counted);	**2**

Assessment Outcomes					
Question	**AO1**	**AO2**	**AO3**	**AO4**	**Question Total**
SECTION A					
1			3a (1)		1
2	1b (1)				1
3	1c (1)				1
4			3d (1)		1
5		2c (1)			1
6.1			3a (1)		1
6.2			3a (1)		1
6.3			3a (2)		2
7.1			3a (1)		1
7.2			3a (1)		1
8				4a/3d (6)	6
9		2e (3)			3
10			3g (3)		3
11				4a (3)	3
12				4a (6)	6
13.1		2b (4)			4
13.2		2b (2)			2
14.1			3e (2)		2
14.2			3e (4)		4
15	1c (6)				6
Total A	**8**	**10**	**17**	**15**	**50**
SECTION B					
16.1			3de (3)		3
16.2		2d (12)			12
17.1			3de (12)		12
17.2			3e (1)		1
17.3			3a (2)		2
Total B	**0**	**12**	**18**	**0**	**30**
Totals	**8**	**22**	**35**	**15**	**80**