Chhetri Academy

chhetriacademy.com

GCSE COMPUTER SCIENCE

2.5

OCR J277 | Specification point 2.5.2

The Integrated Development Environment (IDE)

Untiered

Answers and examiner comments

3	2016-2025	Untiered
question groups	available papers	tier

Using this booklet

1. Answer numbers match the companion question booklet.
2. Follow the original question number on marking tables.
3. Examiner comments are included where available.

Name _____ **Date** _____

Some entries use full source marking material in an appendix. Any unavailable mark schemes are identified beside the relevant answer.

4	a		Sequence	1	
4	b		• A location in memory • The value/contents **cannot** be changed (whilst the program is running)	2	0 mark for "a variable that does not change" 0 marks for "stays the same"
4	c		numberOfPages = numberOfPages+numberOfChapters	1	Accept: • += instead of = numberOfPages • numberOfPages=RoundDown(numberOfWords/wordsPerPage) +numberOfChapters • numberOfPages=RoundDown(numberOfWords/300) +numberOfChapters Variable names must be spelt correctly, ignore case

A451 **Mark Scheme** **June 2016**

Question			Answer/Indicative content	Mark	Guidance
4	d		• Integer/Int • It is a whole number/you can't have half a word	2	Do not allow 'need to ignore the decimal' Cannot get reason if data type incorrect
4	e		• String (name) • Real/Single/Double/Currency/Float/(Decimal) (price)	2	
4	f		1 mark for identification, 1 for matching description e.g. • Error diagnostics/debugger • …highlight errors/suggest changes • Run-time environment • …Lets you run/test the program • Text editor • …highlight key words • …auto-indent • …to type/edit source code • …Auto-complete • …highlight syntax errors • Versioning tools • …To allow for tracing back • …To create new files with changes • Stepping/breakpoints • …Allow tracing of algorithms	4	Do not allow auto-documentation. Can get description mark, without identification/incorrect identification Allow: • Variable watch/window • See how the values change Do not allow compiler/interpreter

A451 **Mark Scheme** **June 2016**

Question			Answer/Indicative content	Mark	Guidance
4	g		Max 2 for compiler, 2 for interpreter Compiler • To convert to low-level in one go • Create an executable//export the file • To distribute the software • Users will have no access to source code... • ...so no-one can edit/steal/copy the code/program • Use for error detection Interpreter • To convert to low-level line by line • To test the program // to find errors • stops running when it finds an error//shows the location of the error when found • it is quicker (compared to compiler) to re-interpret than re-compile	4	The uses must be different for compiler and interpreter

7	**(a)**	**(i)**	1 mark per bullet, max 2. • `else` • `print ("unknown")`	2	Accept logically correct equivalents for else (e.g. `elseif a!="LAN" and/or a !="WAN")`. Do not allow `elseif` on its own Accept other keywords for print (e.g. "output") as long as the intention is clear. Accept other messages as equivalent to "uknown" (e.g. "not known " / "error")) Message to be printed must be in quotes. Allow "`else then`".

Question			Answer	Mark	Guidance
7	(a)	(ii)	1 mark per bullet, max 2. • aimed at humans//understandable by humans / programmers • English like structure / syntax • Must be translated/compiled/interpreted (before it can be run) • Allows programmer to deal with the problem instead of considering the underlying hardware // an abstraction from the hardware // hardware independent // portable	2	Allow examples of keywords (eg IF / ELSE / WHILE) as 2nd bullet point. Do not award marks for naming languages such as Java , Python, etc. Do not award marks for stating what a high level language isn't (i.e. describing what low level code is). Do not allow "easy to use" Do not allow 'has to be converted' without into what i.e machine code etc.
7	(b)		1 mark per bullet, max 4. e.g. • Editor • …to enable **program code** to be entered/edited • Error diagnostics / debugging • …to display information about errors (syntax / run-time) / location of errors • … suggest solutions • Run-time environment • …to enable to the program to be run • … check for run time errors / test the program • Translator / compiler / interpreter • …to convert the high level code into machine code / low level code / binary • ...to enable to code to be executed / run	4	One mark for identifying, one mark for describing. Accept description of a tool without (or with incorrect) naming of the tool. Allow sensible descriptions which go across pairs or name other tools sensibly (e.g. editor / highlighting syntax) Allow any sensible tool that an IDE provides (e.g. auto documentation, help tools, pretty printing etc.)

J276/02 Mark Scheme June 2018

Question			Answer	Mark	Guidance
			• Breakpoints • …to stop/pause program execution at a specific point • Watch window • …to check contents of variables • Stepping • …to execute program line by line • Syntax completion… • …suggests/corrects code • Keyword highlighting / colour coding keywords / pretty printing… • …colours command words / variables		

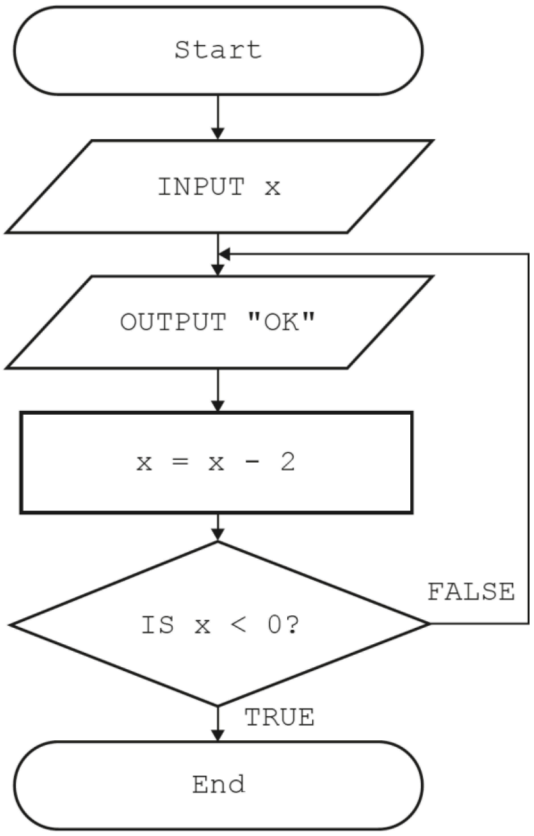
Question			Answer	Mark	Guidance
2	(a)		• 2 • 6	2	
2	(b)	(i)	2 marks per construct max • FOR • Count controlled • repeats a specified number of times • WHILE • Condition controlled • repeats **while** a condition is **true** / **until** a condition is **false** // condition tested before code runs • DO UNTIL / REPEAT UNTIL • Condition controlled • repeats **until** a condition is **true** / **while** a condition is **false** // condition tested after code runs	4	Marks are independent, allow mark for description even if example/type of loop not given. Allow any 2 per construct. Allow suitable examples from other high-level languages. If WHILE and DO UNTIL given, allow “condition controlled” as description of both. Do not give “condition controlled” twice if no other differentiation between types of iteration. Do not allow “repeats infinitely” – this is only true if the condition is never met. Do not allow mismatched answers (e.g. “a while loop repeats until a condition is true”)
2	(b)	(ii)	• Sequence • Selection	2	Allow slight alternatives (e.g. sequencing) Do not allow examples by themselves (e.g. IF) Allow branching as an alternative to selection.

Question			Answer	Mark	Guidance
2	(c)		1 mark for tool 1 mark for matching description e.g. • Text/code editor… • …allows program code to be **written** / **entered / changed** • …allows errors to be fixed • Debugger / error reporting / diagnostics… • …identifies / finds / checks for errors • …shows location / detail of errors • …suggests fixes • Run-time environment / output window… • …allows **program / code** to be run / executed • …shows output of the **program / code** • Translator / compiler / interpreter … • … convert to **low-level/machine code** • …allow program to be executed / run • …produce executable file (only for compiler) • …stops execution when error found (interpreter only) • Stepping … • … execute/run the program line by line	4	Allow other sensible names for features. Description must add more than is given in the identification of the feature to be awarded. For example, “keyword highlighting, highlights keywords” is 1 mark for the feature only. If compiler and interpreter given as two distinct features, allow both (with suitable descriptions). Do not allow translator and compiler/interpreter. **Description must match tool. If tool name wrong / missing, do not give description.** Allow sensible references to AI where appropriate. Allow other sensible features of an IDE (e.g. line numbering, **auto** indent, collapsed blocks, etc) with suitable description. Description must add (slightly) more than just repeating the feature name. Do not allow features of programming languages (comments, casting, etc). For text editor / error diagnostics / debugger, allow other sensible features listed as features in the mark scheme as description (e.g. “text editor does pretty printing”, “debugger does stepping”)

Question			Answer	Mark	Guidance
			• Variable watch… • … see the **contents/data** held in variables / see how (contents of) variables **change** • Break points … • … will allow the program to stop **at a chosen / set position** • Pretty printing // keyword highlighting… • … allows keywords / variables to be coloured / identified • Keyword completion // autocomplete // autocorrect // syntax suggestion… • …suggests/corrects code/syntax (when first part entered). • GUI (builder)… • …allows creation of user interface / buttons / by example		

Question 2 (a)

2 This flowchart uses iteration.



(a) Complete the table by identifying how many times "OK" will be output for when each value is input.

Input value	Number of times "OK" is output
3	
10	

[2]

This question involved following a flowchart to determine the outcome produced. Most students did this successfully for the input value of 3, determining correctly that OK would be output twice.

However, fewer candidates were able to do this for the input value of 10. This may be due to confusion over the use of the 'less than' operator when the value held in variable **x** was set to zero.

Question 2 (b) (i)

(b) Iteration is a programming construct.

(i) Identify **and** describe **two** different examples of iteration that can be used in a high-level language.

1 .....

.....

.....

2 .....

.....

.....

[4]

Iteration is commonly referred to as loops, or repetition. Candidates were confidently able to name examples of keywords that are used in a high-level language (such as **`for`**, **`while`** or **`do until`**) and describe the use of these well. Many candidates used subject-specific terminology such as **condition controlled** loops or **count controlled** loops when doing this, as well as describing these in more depth.

Some candidates lacked precision with their description of a **`while`** loop, stating that this repeats **until** a condition is **`True`** which is incorrect.

Misconception

A **`while`** loop is condition controlled. It has a condition that is evaluated before each iteration begins; if the condition evaluates to be **`True`**, the loop repeats one more time before the condition is evaluated again.

For example, **`while x > 7`** : The condition here that is evaluated is whether the contents of the variable `x` is larger than 7. As long as this is the case, the loop will continue to repeat.

Many candidates suggested that the while loop should repeat **until** the condition is **`True`**, that is, **`while x > 7`** should repeat for values of **`x`** that are not larger than 7 and only stop repeating when **`x`** is larger than 7. This is clearly incorrect.

Centres and candidates must be aware that their answers should be precise and technically correct if they are to achieve highly.

Question 2 (b) (ii)

(ii) Give the names of **two** other programming constructs that are used in a high-level language.

1 .....

2 .....

[2]

Many candidates were able to list sequence and selection as two other programming constructs which are listed in the specification document. We accepted sensible alternatives (such as branching).

Question 2 (c)

(c) A programmer implements the flowchart as a program in a high-level language using an Integrated Development Environment (IDE).

Identify and describe **two** tools that the programmer could use in the IDE to create a program from the flowchart.

Tool 1 .....

Description .....

.....

Tool 2 .....

Description .....

.....

[4]

Candidates were mostly able to name and describe suitable tools contained within an Integrated Development Environment, such as an editor, error diagnostics, run-time environment or translator.

Many other sensible tools and sub-tools such as a variable watch window, stepping, keyword completion or Artificial Intelligence (AI) tools were all also credited.

A small number of candidates mistakenly described tools contained within programming languages, such as commenting or use of variables. Descriptions were generously given where they added to the tool name, but repetition (such as “a code editor / edits code”) was not given both mark parts.