2.10 Translators and Integrated Development Environments (IDEs): 30-minute Pack

OCR J277 · Calculators not allowed · about 30 minutes

Total Marks

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 Programs can be written in either a high-level or a low-level language.

(a) State one advantage of writing a program in a high-level language rather than a low-level language. **(2)**

(b) State one advantage of writing a program in a low-level language rather than a high-level language. **(2)**

(Total for Question 1 is 4 marks)

2 Explain how an interpreter translates and executes a program, including what happens if it reaches a line containing a syntax error.

(Total for Question 2 is 4 marks)

© Revision Library | revisionlibrary.co.uk

3 A junior programmer keeps forgetting to close brackets and does not indent the inside of loops consistently, making the code hard to read. Identify two IDE tools that would help and explain how each one addresses the problem.

(Total for Question 3 is 4 marks)

4 A software company is deciding whether to develop a new mobile game using a compiler or an interpreter. Explain the key differences between compilers and interpreters, and discuss which would be more appropriate for this mobile game, justifying your answer.

(Total for Question 4 is 6 marks)

© Revision Library | revisionlibrary.co.uk

5 An Integrated Development Environment (IDE) is a piece of software that provides tools to help a programmer write, test and debug programs.

(a) Which IDE tool shows different types of code element (e.g. keywords, strings, comments) in different colours, to make code easier to read? **(1)**

- ☐ A) Auto-complete
- ☐ B) Syntax highlighting
- ☐ C) Breakpoint
- ☐ D) Run-time environment

(b) Which IDE tool automatically suggests the rest of a keyword, variable or function name as the programmer types, to save time and reduce spelling mistakes? **(1)**

- ☐ A) Auto-complete
- ☐ B) Error diagnostics
- ☐ C) Translator
- ☐ D) Debugger

(c) Which IDE tool checks code for syntax errors as it is typed, underlining and reporting the location and type of any errors found? **(1)**

- ☐ A) Auto-complete
- ☐ B) Error diagnostics
- ☐ C) Breakpoint
- ☐ D) Refactoring tool

(d) Which IDE tool lets a programmer run a program from within the IDE, without needing separate software, so it can be tested immediately? **(1)**

- ☐ A) Run-time environment
- ☐ B) Version control
- ☐ C) Auto-indentation
- ☐ D) Stepping

(Total for Question 5 is 4 marks)

© Revision Library | revisionlibrary.co.uk

6 Discuss how IDE tools support a programmer while developing and testing a large program, referring to at least three different IDE tools in your answer.

(Total for Question 6 is 6 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 2.10 Translators and Integrated Development Environments (IDEs): 30-minute Pack

Question 1

(a) easier/quicker for a programmer to write and understand, since the syntax is closer to natural English, oe — **B1**

(a) more portable - the same source code can often be compiled/run on different types of computer with little or no change, oe — **B1**

(a) Answer: It is easier and quicker to write and understand, and is more portable across different types of computer.

(b) runs faster/more efficiently, since it does not need to be translated into a less direct set of instructions, oe — **B1**

(b) gives the programmer precise control over exactly how the hardware/memory is used, which matters for embedded systems or maximum performance, oe — **B1**

(b) Answer: It runs faster/more efficiently and gives the programmer precise control over the hardware.

Question 2

the interpreter translates and executes the source code one line/statement at a time, oe — **B1**

each line is checked and run before the next line is translated, oe — **B1**

if a line contains an error, the interpreter stops immediately at that line and reports the error, oe — **B1**

any lines already executed before the error was reached will already have had their effect, but no executable file is produced/saved, oe — **B1**

Answer: The interpreter translates and runs each line in turn; if it reaches a line with an error it stops immediately and reports it, and no executable file is ever saved.

Question 3

one suitable IDE tool, e.g. auto-indentation/code formatting; bracket matching/syntax highlighting; automatic bracket completion; error diagnostics, oe — **B1**

a linked explanation of how that tool corrects inconsistent indentation or helps identify/prevent an unclosed bracket, oe — **B1**

a second, different suitable IDE tool, oe — **B1**

a linked explanation of how that second tool addresses a stated problem, oe — **B1**

Answer: For example: auto-indentation or code formatting keeps loop bodies consistently indented; bracket matching, automatic bracket completion or error diagnostics helps identify or prevent unclosed brackets. Accept two suitable tools with matching explanations.

Question 4

Level 1 Basic, undeveloped comments about compilers and/or interpreters are given, with little or no reference to the mobile game scenario. Little relevant technical terminology is used. — **1-2**

Level 2 At least one genuine difference between compilers and interpreters is explained with some understanding, and a choice is made for the mobile game, but the justification may be one-sided or lack detail in places. Relevant technical terminology is generally used correctly. — **3-4**

Level 3 A well-balanced discussion clearly explains multiple differences between compilers and interpreters (e.g. speed, error reporting, distribution) and reaches a justified, well-supported conclusion for the mobile game scenario. Relevant technical terminology is used correctly throughout. — **5-6**

Indicative content:

- A compiler translates the whole program before it is run, producing a standalone executable; an interpreter translates and executes line by line and needs the interpreter present every time the program runs
- Compiled programs generally run faster/more efficiently, which matters for a real-time mobile game that needs smooth performance
- Compiling produces one distributable file that does not expose the readable source code, which also helps protect the company's work from being copied
- A compiler reports all syntax errors together before anything runs, so a fully tested, compiled game is less likely to crash from small errors once released
- An interpreter can be useful earlier in development, for testing small sections of code quickly, but this matters less for the final released product
- A justified conclusion should recommend a compiler for the finished, released mobile game, since running speed and having a single protected executable matter more than line-by-line testing at this stage

Question 5

(a) B cao — **B1**

(a) Answer: B) Syntax highlighting

(b) A cao — **B1**

(b) Answer: A) Auto-complete

(c) B cao — **B1**

(c) Answer: B) Error diagnostics

(d) A cao — **B1**

(d) Answer: A) Run-time environment

© Revision Library | revisionlibrary.co.uk

Question 6

Level 1 One or two IDE tools are named, with little or no explanation of how they help. Little relevant technical terminology is used. **1-2**

Level 2 At least two IDE tools are named and explained with some understanding of how each helps a programmer, but the discussion may lack detail or fail to link the tools together. **3-4**

Level 3 Three or more IDE tools are clearly named and explained (e.g. from editing, error-checking and debugging categories), with a coherent discussion of how, together, they reduce development time and the number of errors in a large program. Relevant technical terminology is used correctly throughout. **5-6**

Indicative content:

- A source code editor with syntax highlighting, auto-indentation and auto-complete helps a programmer write code faster and with fewer typing mistakes
- Error diagnostics/real-time syntax checking catches mistakes as they are typed, before the program is even run
- A built-in run-time environment lets the programmer test the program immediately, without needing separate software
- Breakpoints, single-stepping and variable/watch windows let a programmer pause execution and inspect variable values, helping locate the exact point a logic error occurs
- Refactoring tools let a programmer safely restructure code (e.g. renaming a variable everywhere it is used) across a large program without missing an occurrence
- Together, these tools reduce development time and the number of errors that reach the finished program, which matters more as a program becomes large and complex

© Revision Library | revisionlibrary.co.uk