NAME DATE STARTED TARGET END DATE

GCE A LEVEL — COMPUTER SCIENCE UNIT 3 QUESTION PACK

1500U30-1 · 2015 spec Unit 3 Topic 7 · A2 unit, first sat 2017, 100 marks, 2h paper

REVISE.wales

COMPUTER SCIENCE — UNIT 3 · Compilers, Interpreters & the IDE

Topic 3.7 — Compilation pipeline, assemblers, debugging tools, version management and code editors

The lexical, syntax and semantic analysis stages of compilation, advantages and drawbacks of compiled vs interpreted languages, the role of an assembler, IDE features for writing and editing code (syntax highlighting, auto-complete), debugging tools (stepping, breakpoints, variable watches), translation vs execution errors, and program version management.

2015 specification · current

Estimated time for entire question pack: ~1 h 46 min

Derived from the Unit 3 pace of ~1.5 min/mark, padded for written-prose answers (71 marks over 8 questions).

*You are advised to **not** attempt to complete all of this in one sitting.*

ABOUT THIS QUESTION PACK

This is a **comprehensive topic question pack**, not a single mock paper. It contains every question from the WJEC A2 Unit 3 papers (Summer 2017 – Summer 2024, COVID gap) that maps onto Topic 3.7 of the 2015 specification.

Questions are ordered by source paper date.

INSTRUCTIONS

Use black ink or black ball-point pen. Show all working. A calculator is allowed where useful.

For Examiner's use only

Q	Source	Max	Mark
1	S17 Q4	16	
2	S17 Q12	6	
3	S18 Q5	9	
4	S19 Q13	13	
5	S23 Q8	9	
6	S23 Q10	4	
7	S23 Q11	8	
8	S24 Q8	6	
Total		**71**	

Compilers, Interpreters & the IDE — what the spec asks

WJEC GCE A Level Computer Science (from 2015) · Unit 3: Programming & System Development · Topic 3.7.

Compilation pipeline

- Lexical analysis: source → tokens (keywords, identifiers, literals).
- Syntax analysis (parsing): tokens → parse / abstract syntax tree (grammar check).
- Semantic analysis: type checking, scope resolution, undefined-variable check.
- Code generation + optimisation: emit machine code or intermediate code.

Compiler vs interpreter

- Compiler: translates whole program to machine code before running; faster execution.
- Interpreter: executes source line-by-line; slower but easier debugging and platform-independent.
- Compiled examples: C, C++, Rust, Go.
- Interpreted examples: Python, JavaScript, Ruby, BASIC.

Assemblers

- Translate assembly source code (one mnemonic per machine instruction) to machine code.
- Source code uses mnemonics (MOV, ADD) and labels — mostly 1:1 with machine code.
- Contrast: a high-level statement usually maps to many machine instructions.
- Used when fine-grained hardware control or maximum performance is required.

Code editor & IDE features

- Syntax highlighting: colours keywords, strings, comments for readability.
- Auto-complete / IntelliSense: suggests identifiers as you type.
- Bracket matching, code folding, indentation aids.
- Built-in compiler / interpreter, integrated terminal.

Debugging tools

- Stepping: execute one instruction at a time (step into, step over, step out).
- Breakpoints: pause execution at a chosen line.
- Variable watches: monitor variable values as the program runs.
- Helps find logic errors that translation can't detect.

Errors & version control

- Translation (syntax / lexical) errors: prevent compilation, e.g. missing semicolon, mismatched bracket.
- Execution / runtime errors: surface while running, e.g. divide by zero, null reference.
- Logic errors: program runs but produces wrong output; only debugging finds them.
- Version management (Git, SVN) tracks history, supports branching, rollback, code review.

Compilers, Interpreters & the IDE in one page

Quick-reference notes — revisit before each question.

Compilation stages

1. Lexical → tokens.
2. Syntax (parse) → parse tree.
3. Semantic → type and scope checks.
4. Code generation → machine / intermediate code.

Optimisation can happen at multiple stages.

Compiler vs interpreter

Compiler: whole program once, fast at run.
Interpreter: line-by-line, slower but interactive.
Compiled: C, C++, Go, Rust.
Interpreted: Python, JavaScript, Ruby.

Assembler

Translates assembly (mnemonics) → machine code.
Mostly 1:1 mapping; very low-level.
Compare with compiler input (high-level source), which is many-to-one.

IDE editor features

Syntax highlighting.
Auto-complete / IntelliSense.
Bracket / paren matching.
Code folding.
Inline error squiggles.

Debugger tools

Stepping: step over / into / out.
Breakpoints: pause execution at a line.
Variable watches: live variable values.
Stack trace: how you got here.

Error categories

Lexical: bad token (illegal character).
Syntax: wrong grammar (missing ;, mismatched bracket).
Semantic: type mismatch, undeclared variable.
Runtime: division by zero, null deref.
Logic: wrong output, no error message.

Answer **all** *questions.*

1. Two human computer interfaces (HCI) are voice input and touch screen.

(a) Give **two** benefits of using a touch screen interface on a mobile device. [2]

(b) Describe the difficulties in creating a natural language interface for voice input. [4]

2. Explain the terms object, class and method in object-oriented programming. [3]

3. *(a)* Draw a truth table to show the value of **P** for all possible values of **A**, **B** and **C** for the following Boolean expression:

$$P = \bar{A} \cdot B + C$$ [4]

(b) Using the data in the 8 bit register below, design a mask and use it to demonstrate how a logical operation can be used to extract the least significant 4 bits. [3]

Bit number	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Register contents	1	1	1	0	0	0	1	1

4. *(a)* Describe the processes carried out during the lexical, syntax and semantic analyses stages of compilation. [6]

(b) Describe **one** advantage of using a programming language that requires compiling compared with a programming language that requires interpreting. [2]

(c) Describe **two** advantages of using a programming language that requires interpreting compared with a programming language that requires compiling. [4]

(d) State the purpose of an assembler. Describe the difference between the source code of an assembler and the source code of a compiler. [4]

5. *(a)* Using the laws of Boolean algebra and De Morgan's theorem simplify the following Boolean expression:

$$A \cdot (\overline{A \cdot B})$$ [4]

(b) Simplify the following Boolean expression:

$$A \cdot B + A \cdot (B + C) + B \cdot (B + C)$$ [4]

6. Write a bubble sort algorithm, using pseudo-code, to sort an array of integers into ascending order. [9]

7. The name of a constant in a certain computer language must either be a single uppercase letter, or a single uppercase letter followed by one or more uppercase letters or digits.

(a) Produce an appropriate syntax diagram to define a constant in this language. [4]

(b) Produce an appropriate Backus-Naur Form (BNF) definition for a constant in this language. [4]

8. Below is part of an algorithm that initialises a two dimensional array named GRID of size N x N with zeros.

```
for i = 0 to N - 1
          for j =  0 to N - 1
                GRID(i,j) = 0
          next j
    next i
```

Evaluate the efficiency of the algorithm and, using Big O notation, determine the growth rate for the time performance. [4]

9. *(a)* Define the term algorithm. Other than pseudo-code, state **one** method of expressing an algorithm. [2]

(b) Describe the main characteristics of a recursive algorithm. [2]

(c) Describe **two** disadvantages of using a recursive algorithm. [2]

10. A software company carries out a design review as part of its quality control procedures.

Identify **three** tasks that would be carried out during a design review. [3]

11. A linked list can be ordered or unordered.

(a) Explain the difference between searching for an item in an ordered list compared with searching an unordered list. [2]

(b) The table below includes an unordered list of names.

Index	Data	Next Pointer (1)	Next Pointer (2)	Next Pointer (3)
0	Smith			
1	Jones			
2	Ahmed			
3	Lewis			
4	Thomas			
5	Brown			
6				
7				
8				
9				

(i) Copy the table and complete the **Next Pointer (1)** column to link the list in ascending alphabetical order. [3]

(ii) Add Murphy and Collins to the linked list and complete the **Next Pointer (2)** column. [4]

(iii) Complete the **Next Pointer (3)** column to delete Smith. [2]

(c) Draw a representation of a binary tree using the data items from question **11***(b)* as key values. [3]

12. A debugging tool of an Integrated Development Environment (IDE) enables stepping, break points and variable watches.

Describe the use of stepping, break points and variable watches in the debugging of programs. [6]

13. Compare the Waterfall and Agile approaches to systems analysis and discuss suitable programming paradigms for each approach. [10]

END OF PAPER

5. A code editor is an essential software engineering tool of an integrated development environment (IDE). It provides features designed to assist with the writing and editing of code.

(a) Describe **two** features of a code editor designed to assist with the writing and editing of code. [4]

(b) An IDE will also include tools for translation and debugging of the code.
State the purpose of code translation. [1]

(c) Errors in the code will stop the program from working as intended. Some errors will prevent code translation, other errors may not be discovered until the program is in use.
Describe, giving examples, an error that will prevent program translation and an error that will not be discovered during translation. [4]

13. Describe the purpose, and give examples of, the use of compilers, interpreters and assemblers and distinguish between them. [13]

END OF PAPER

4. Clearly showing each step, simplify the following Boolean expressions using Boolean algebra, identities and De Morgan's Law where appropriate.

(a) $A.(1+C)+\overline{B}.(A+B)$ [5]

(b) $X.\left(\overline{Y+Z}\right)+\overline{Z}.X$ [5]

5. Write a quicksort algorithm, to sort an array of integers into ascending order. [8]

6. State the purpose of validation and verification, giving a suitable method for each. [4]

7. An online grocery store uses binary trees. These binary trees can be traversed using a variety of methods.

(a) Describe the following methods of traversal and give an example of why each method would be used in the grocery store.

(i) In-order traversal [3]

(ii) Post-order traversal [3]

(iii) Pre-order traversal [3]

(b) Draw an example of a balanced binary tree for grocery items. [1]

8. Giving suitable examples, describe the types of software tool that have been designed to assist in the following:

(a) Analysis and planning [3]

(b) Software development [3]

(c) Version management [3]

9. A QR code generator uses a string to produce a WiFi access QR code. The string comprises the QR code type, encryption type, network name (SSID) and password. The string is separated using colons.

- The QR code type is WIFI.
- Colon (:)
- Encryption can be either WEP or WPA.
- The network name can contain only lowercase letters, uppercase letters and digits.
- The password can contain lowercase letters, uppercase letters, special characters and digits.

Example: WIFI:WEP:WJECWiFiPublic:Pa$$w0rd1

Produce a Backus-Naur Form (BNF) definition for the string. [5]

10. Giving suitable examples, describe the differences between translation and execution errors. [4]

11. Describe the difference between compilers and interpreters. Give one example of a language which is compiled and one example of a language which is interpreted. [8]

9. A QR code generator uses a string to produce a WiFi access QR code. The string comprises the QR code type, encryption type, network name (SSID) and password. The string is separated using colons.

- The QR code type is WIFI.
- Colon (:)
- Encryption can be either WEP or WPA.
- The network name can contain only lowercase letters, uppercase letters and digits.
- The password can contain lowercase letters, uppercase letters, special characters and digits.

Example: WIFI:WEP:WJECWiFiPublic:Pa$$w0rd1

Produce a Backus-Naur Form (BNF) definition for the string. [5]

10. Giving suitable examples, describe the differences between translation and execution errors. [4]

11. Describe the difference between compilers and interpreters. Give one example of a language which is compiled and one example of a language which is interpreted. [8]

9. A QR code generator uses a string to produce a WiFi access QR code. The string comprises the QR code type, encryption type, network name (SSID) and password. The string is separated using colons.

- The QR code type is WIFI.
- Colon (:)
- Encryption can be either WEP or WPA.
- The network name can contain only lowercase letters, uppercase letters and digits.
- The password can contain lowercase letters, uppercase letters, special characters and digits.

Example: WIFI:WEP:WJECWiFiPublic:Pa$$w0rd1

Produce a Backus-Naur Form (BNF) definition for the string. [5]

10. Giving suitable examples, describe the differences between translation and execution errors. [4]

11. Describe the difference between compilers and interpreters. Give one example of a language which is compiled and one example of a language which is interpreted. [8]

6. When evaluating computer-based solutions, there are several criteria that can be considered.

Describe the following criteria when evaluating computer-based solutions and give suitable examples:

(a) Usability [2]

(b) Performance [2]

(c) Scalability [2]

(d) Security [2]

7. A website URL is made up of a protocol, a domain name, and an optional file path.

- The protocol can only be “http” or “https”.
- The domain name can only consist of alphanumeric characters, hyphens and full stops.
- The protocol and domain name must be separated by a colon and two forward slashes.
- The optional file path must start with a forward slash and can only contain alphanumeric characters and forward slashes.

Example: https://www.wjec.co.uk/home/

Produce a Backus-Naur form (BNF) definition for a valid website URL. [6]

8. Explain program version management. [6]

9. Explain, using suitable examples, recursive and non-recursive sorting algorithms. [8]

END OF QUESTION PACK

8 questions · 71 marks · ~1 h 46 min
Source: WJEC A2 Computer Science Unit 3 (1500U30-1), Summer 2017–2024, COVID gap
Curated for WJEC Computer Science 2015 spec A2 Unit 3 — Topic 7 (3.7)