|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **Computer Science (9-1)**  Translators and IDEs Paul Burgess |  | | Please note that you may see slight differences between this paper and the original.  Candidates answer on the Question paper.  **OCR supplied materials:** Additional resources may be supplied with this paper.  **Other materials required:** •   Pencil •   Ruler (cm/mm) | **Duration:** Not set | |  | | |  |

## INSTRUCTIONS TO CANDIDATES

•   Write your name, centre number and candidate number in the boxes above. Please write clearly and in capital letters.  
•   Use black ink. HB pencil may be used for graphs and diagrams only.  
•   Answer **all** the questions, unless your teacher tells you otherwise.  
•   Read each question carefully. Make sure you know what you have to do before starting your answer.  
•   Where space is provided below the question, please write your answer there.  
•   You may use additional paper, or a specific Answer sheet if one is provided, but you must clearly show your candidate number, centre number  
    and question number(s).

## INFORMATION FOR CANDIDATES

•   The quality of written communication is assessed in questions marked with either a pencil or an asterisk. In History and Geography   
    a *Quality of extended response* question is marked with an asterisk, while a pencil is used for questions in which *Spelling, punctuation and  
    grammar and the use of specialist terminology* is assessed.  
•   The number of marks is given in brackets **[ ]** at the end of each question or part question.  
•   The total number of marks for this paper is **32**.  
•   The total number of marks may take into account some 'either/or' question choices.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **1(a).** | Harry is planning to create a computer game using a high-level programming language.  State why the computer needs to translate the code before it is executed.     |  | | --- | | **[1]** | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **(b).** | Harry can use either a compiler or an interpreter to translate the code.  Describe **two** differences between how a compiler and an interpreter would translate Harry's computer game.                   |  | | --- | | **[4]** | | | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **2.** | The area of a circle is calculated using the formula Π × r2, where Π is equal to 3.142 and r is the radius.  Finn has written a program to allow a user to enter the radius of a circle as a whole number, between 1 and 30, and output the area of the circle.    Finn uses an IDE (Integrated Development Environment) to write his programs. Identify **two** features of an IDE that Finn might use.      **[2]** | | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **3.** | Graeme is a freelance programmer. He has written a program for a client and gives the client both the high level code and the machine code of the program.   1. Describe what is meant by  High level code               Machine code               1. State why Graeme needs a compiler.         **[1]** | | |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **4(a).** | Joseph is an author and programmer, and he needs to estimate how many pages his new book will have.  Each page has an average of 300 words. Each chapter starts with a chapter title page. The number of pages is estimated by;   * dividing the number of words by 300 * ignoring the decimal part of the division * adding the number of chapters to this total.   Joseph uses the algorithm below to estimate the number of pages, but his algorithm does not give the correct result.   |  |  | | --- | --- | | 01 | INPUT numberOfWords | | 02 | INPUT numberOfChapters | | 03 | CONST wordsPerPage = 300 | | 04 | numberOfPages = RoundDown(numberOfWords / wordsPerPage) | | 05 | numberOfPages = numberOfWords + numberOfChapters | | 06 | OUTPUT numberOfPages |   Joseph has used a RoundDown function to remove the decimal part of the division, e.g. RoundDown(6.2) would return 6, RoundDown(7.8) would return 7.  Joseph is using an Integrated Development Environment (IDE) to produce the program.  One tool in an IDE that Joseph uses is a translator.  Describe **two** additional tools in an IDE that Joseph could use to help him produce his program.   |  |  | | --- | --- | | Tool 1 name: |  |  |  |  | | --- | --- | | Tool 1 description: |  |      |  |  | | --- | --- | | Tool 2 name: |  |  |  |  | | --- | --- | | Tool 2 description: |  |     **[4]** | |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **(b).** | Joseph's IDE allows him to use both a compiler and an interpreter.  Describe how Joseph could make use of a compiler and an interpreter when producing his program.   |  |  | | --- | --- | | Compiler: |  |          |  |  | | --- | --- | | Interpreter: |  |         **[4]** | | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **5.** | Victoria creates a program using an Integrated Development Environment (IDE).  Describe two tools or facilities that an IDE commonly provides.            **[4]** | | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **6.** | Jim is writing a program to calculate the wages of workers in a teddy bear factory.  Jim uses an Integrated Development Environment (IDE) to create the program.  Describe **two** tools in an IDE that can help Jim when creating the program.  1          2          **[4]** | | |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  |  | | **7.** | Victoria is writing a program using a high level language to display the meaning of computer science acronyms that are entered. The code for her first attempt at this program is shown below.   |  |  |  |  | | --- | --- | --- | --- | |  | 01 | a = input(“Enter an acronym”) | | |  | 02 | if a == “LAN” then | | |  | 03 |  | print(“Local Area Network”) | |  | 04 | elseif a == “WAN” then | | |  | 05 |  | print(“Wide Area Network”) | |  | 06 | …………………………………………………………………………………………… | | |  | 07 | …………………………………………………………………………………………… | | |  | 08 | endif | |  |  |  |  | | --- | --- | --- | | **(i)** | Complete the code above to print out an “unknown” message if any other acronym is entered by the user. | **[2]** | | **(ii)** | Describe what is meant by a “high level language”. | | |  |  | | |  |  | | |  | **[2]** | | | | |

**END OF QUESTION paper**

# Mark scheme

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Question** | | | **Answer/Indicative content** | **Marks** | **Guidance** |
| 1 | a |  | * To convert it to binary / machine code * The processor can only understand machine code | 1 | Maximum 1 mark |
|  | b |  | * Compiler translates all the code in one go… * … whereas an interpreter translates one line at a time * Compiler creates an executable… * …whereas an interpreter does not / executes one line at a time * Compiler reports all errors at the end… * … whereas an interpreter stops when it finds an error | 4 | 1 mark to be awarded for the correct identification and one for a valid description up to a maximum of 4 marks. No more than 2 marks for answers relating only to interpreters and no more than 2 marks for answers only relating to compilers. |
|  |  |  | **Total** | **5** |  |
| 2 |  |  | * Error diagnostics (any example) * Run-time environment * Editor (any feature such as auto-correct, auto-indent) * Translator * Version control * Break point * Stepping | 2 | 1 mark per bullet to a maximum of 2 marks.Only 1 example per bullet, e.g. auto-correct and auto-indent would only gain 1 mark. |
|  |  |  | **Total** | **2** |  |
| 3 |  | i | High level code :   * human oriented code / written by programmers * contains words for commands / closer to English / natural language * Machine independent / Portable to different systems * Needs to be translated before it can be executed. * Problem based * One (high level) command equates to many machine code instructions.   Machine code:   * Code for the CPU to execute / not readily understandable by humans * binary instructions * specific to a particular (type of) computer / not portable to different systems * does not need to be translated   [max 2 marks for each type of code] | 4 | Award marks for correct points about machine code made under high level code and vice versa.  Do not accept Machine code is in Hex |
|  |  | ii | * To translate the high level code into machine code * To pick up (syntax) errors | 1 | Translate to object code is acceptable Accept “errors” on its own, but do not accept answers referring specifically to logic or runtime errors. |
|  |  |  | **Total** | **5** |  |
| 4 | a |  | 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  **Examiner's Comments**  This question was appropriate programming theory and techniques.  This question was not answered well. Many candidates did not know what an IDE was, often giving utility programs as responses. A significant number of candidates gave compiler and interpreter as answers, showing that they did not understand that these are examples of the translator that was given in the question. The better candidates were able to name the tools, as given in the specification, and describe them. Marks were often given for descriptions of the tools, where they could not be named. |
|  | b |  | 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 recompile | 4 | The uses must be different for compiler and interpreter  **Examiner's Comments**  This question was appropriate programming theory and techniques.  Many candidates did not answer the question, instead giving definitions of compilers and interpreters, instead of describing how they were used when producing a program. The most common answers involved checking for errors. |
|  |  |  | **Total** | **8** |  |
| 5 |  |  | 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 * 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 | **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.)        **Examiner’s Comments**  Most candidates were able to gain at least some marks from this question, with a significant number able to access all 4 marks. A list of common tools and facilities available in an IDE is given in the specification (section 2.5 ‘Translators and facilities of languages’) although other sensible tools that candidates may have had experience with were also accepted. Although many candidates were able to name the tools, they often struggled to describe these and give more comprehensive answers. Again, extensive practical use of a high-level language programming language alongside an IDE would have been extremely beneficial to candidates and centres. |
|  |  |  | **Total** | **4** |  |
| 6 |  |  | e.g.   * Editor * Allows Jim to enter the program code * Colour coding keywords * Auto-completes code as you type. * Compiler * Transforms the written source code into machine code. * Debugging tools * Highlights errors in the code * Suggests possible solutions.   (2 marks per tool) | 4 | Do not accept me spell check  **Examiner's Comments** It was pleasing to see that candidates were using their experience of programming to answer the question. Where they did not gain full marks, this could have improved by providing further detail about the tools, or especially by using correct technical terms to describe these tools. |
|  |  |  | **Total** | **4** |  |
| 7 |  |  | 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”.   **Examiner’s Comments**  Most candidates were able to complete line 07 successfully, with an output/print of an acceptable ‘unknown’’ message being all that was required. Line 06 required candidates to understand that the message on line 07 should be printed out where the conditions in lines 02 and 04 were both false; the simplest way of achieving this was an ELSE (or equivalent). Where candidates put more logically complex statements, these were successful if they were logically correct. However, many candidates struggled with this.  **Misconception** To compare the contents of a variable against two possible values, it is incorrect to use :   •  if a!=‘WAN’ or ‘LAN’ In the example above, the comparison of a against ‘WAN’ is logically correct, but it is unclear what ‘LAN’ is being compared against. A logically correct way to achieve the same thing would be :   •  if a!=‘WAN’ and a!=‘LAN’ |
|  |  |  | 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.   **Examiner’s Comments**  This question was answered well by many candidates, with responses relating to the use of English-like keywords but needing to be translated before the processor could execute it being the most popular. A minority of candidates confused ‘high-level’ with ‘difficult’ and gave incorrect answers regarding it being too hard to programmers to use. The opposite is in fact true, with ‘high-level’ referring to the level of abstraction away from the underlying hardware.  Very few answers were seen that in any way discussed this abstraction or portability between different processors. Hopefully centres will become more confident with the delivery of the subject and so more complex and technically complete answers will be seen across all questions from high ability candidates. |
|  |  |  | **Total** | **4** |  |