10(a)	**1 mark** per bullet point, **max 4** marks **Max 2** from each section. Interpreter: • Use an interpreter while writing/coding the program • … to test / debug the partially completed program • … because errors can be corrected and processing continue from where the execution stopped // errors can be corrected in real time // errors are identified one at a time Compiler: • Use the compiler after the program is complete • … to create an executable file so source code not seen • Use the compiler to repeatedly test the same completed section without having to re-interpret every time
10(b)	**1 mark** per bullet point, **max 2** • The software is free of restrictions • The software is not necessarily free of charge • The source code comes with the program // A user can edit the source code to add functionality / suit their needs • Users can share the software with others
10(c)	**1 mark** for a correct answer • Digital Signature

8(a)	**1 mark** per bullet point, **max 2** • Large amounts of source code take time to compile • It can be slower to produce the object code than an interpreter • The code must be recompiled when it is changed • The program cannot run if there are errors • It is not possible to correct errors in real-time • One error can cause false reporting of multiple further errors • Sections of code / unfinished code cannot easily be tested
8(b)	**1 mark** per bullet point**, max 3** marks • Programming time is saved as code does not have to written from scratch • Testing time is saved as code is already tested / documented • A library routine is more likely to work, as code is already tested • Library routines automatically update if they are changed / improved • The programmer can use library routines to perform complex functions / procedures that they may not be able to write themselves
8(c)	**1 mark** per bullet point**, max 2** marks • Open Source (Initiative) • Free Software (Foundation)
8(d)	**1 mark** for each bullet point, **max 3** marks • File management • Security management • Hardware / peripheral management // input / output management • Process management

3(a)	**1 mark** for each correctly completed term • **an executable file/.exe** • **source/program code** • **stops** • **immediately/in real-time** A compiler checks all of the code before attempting to translate the program. If any errors are found, they are all reported at the same time and the program does not translate or run. If there are no errors found, the compiler produces **an executable file/.exe** which can run without access to the **source/program code**. An interpreter translates one line of code and then runs it, before moving to the next line of code. If the line of code has an error, the interpreter **stops** and displays the error. The programmer can correct the error **immediately/in real-time** and then the interpreter continues translating from that point.
3(b)	**1 mark** for method and **1 mark** for corresponding description During transmission e.g. • Encryption // by example such as VPN • Jumble / encode data so it cannot be decrypted/understood without the key On computer e.g. • Firewall / proxy • Filter incoming transmissions and stop any that could be attempting unauthorised access • Anti-malware • Find and delete or quarantine any malware that could delete the data / files • Encryption • Jumble / encode data so it cannot be decrypted / understood without the key • Physical method // by example • For example, the computer storing the data cannot be accessed without the key to the room

3(c)	**1 mark** each to **max 4** • A webserver stores all the data for each player • Each player is on a client computer // The player's web browser is the client • that sends requests over the internet to the web server • the server performs the required action in the game • the server updates the data in the game • the server sends the results to the player
3(d)	**1 mark** each to **max 3** e.g. • There is no / limited access to legal advice • in case action is taken against them • There are fewer networking opportunities • so they could miss out on contacts / jobs • There would be less access to training • There would be no clear laid out ethical guidelines • and / or people to discuss potential ethical problems with • possibly leading to inappropriate / unethical actions • which might lead to legal proceedings / recourse

4(a)	**1 mark** for each bullet point (**max 4**) e.g. • Single stepping • Run the program one line at a time • … and check the variable contents / program flow // show the effect of each line of code • Set breakpoints • … run the code up to a set line • ... and then check the status • Variable/report watch window • … view how the data changes as the program is running
4(b)	**1 mark** for each bullet point (**max 3**) e.g. • Subroutines can be shared / reused • … between team members who are working independently • … without having to rewrite/re-test them which saves the programmers' time • A program library provides continuity between programs/programmers • Individual programmers can contribute their specialisms to the library // Individual programmers can use the specialisms of others
4(c)	**1 mark** for the security method. **2 marks** for explanation Security method: Encryption Explanation • File contents are converted to cipher text • If intercepted the data cannot be understood without the decryption key

3(a)	**1 mark** each to **max 2**: • The interpreter will stop when an error is found • ... so the error can be corrected in real-time, and the result of changes seen immediately • Only one error is displayed at a time • … so fewer errors to correct simultaneously **and** no dependent errors
3(b)	**1 mark** each to **max 3**: • Program can be distributed without source code • ... so it cannot be edited/stolen/plagiarised • Users do not require the translator to run the program • … so time is not spent retranslating by user

8(a)	**1 mark** each to **max 2**: • Creates an executable file • … so the code can be tested multiple times without having to recompile • … so **repeated testing** takes less time
8(b)	**1 mark** for identification of each feature and **1 mark** for matching description: e.g. *For coding:* **IDE feature** / **Description** Context-sensitive prompts / **Gives suggestions** for code as the user types instead of having to write/remember the code **Auto**-correct / Corrects spelling mistakes so that user has fewer errors to correct *For presentation:* **IDE feature** / **Description** Pretty-printing / Colour code keywords so the user can identify any errors Expand/collapse (code) blocks / The user can hide code that they are not currently working on *For debugging:* **IDE feature** / **Description** Single stepping / Run the code **one line at a time** // shows the effect of each line of code Breakpoints / Stop the code running at a set point to check the flow/variable contents
8(c)	**1 mark** each to **max 2**: • Saves programming/testing time as code does not have to be written/re-written from scratch // code does not have to be tested • Code is already tested so it is more robust/likely to work • The programmer does not need to maintain the library // library routines are updated automatically • Can perform complex calculations that the programmer may be unable to do • Makes code more easily readable