BCS Higher Education Qualification

Certificate in IT

April 2023

EXAMINERS' REPORT

Software Development

Questions Report:

A1	
	This question had four parts. Parts a) and b) applied coding practices whereas parts c) and d) covered concepts of functions and procedures. Part a) required candidates to understand the supplied algorithm and then to express the algorithm as a sequence of steps that traced the processing required to convert a 10 (in decimal) to the equivalent binary number. Many candidates attempted this part using a trace table identifying the need to iterate through each step. However many candidates lost marks by simply repeating the algorithm with little detail. Some candidates included a flow chart that was not required but may have assisted candidates understanding of the problem. Part b) required candidates to write code using their answers to part a) to guide them. Candidates exhibited a wide range of coding however there was an inability to accurately express iteration selection statements and a lack of understanding integer division. Furthermore, many candidates lost marks because they did not use a function required to convert decimal to binary. Instead all the code was contained in one main program. The main program should prompt the user for input; call the function with the input decimal value and then output the result as a binary string in reverse order. Most candidates used an array as the data structure to store the result as expected. A small number used a stack instead of an array. Parts c) and d) - many answers showed a basic knowledge of functions. However it was pleasing to see candidates supply example code to illustrate the essential differences between functions and procedures. It is worth noting the use of built in functions such as those provided in Python. Sometimes these can be excessive and should be avoided if native code can be expressed by the candidate instead. This question concerned an algorithm to convert decimal to binary. Part a) was related to tracing steps in the algorithm. A significant number of candidates gained marks for correctly tracing each division and remaining steps, but failed to correctly assign the array values. For part b) it was necessary to implement the algorithm as a procedure or function. Many candidates failed to implement the full algorithm or did not provide the code for calling the function or procedure. Parts c) and d) were difficult for many candidates with few giving a good account of differences between them. Very few mentioned the essential difference as a call by reference allowing the possibility of the argument being changed.

A2	
	This was a popular question with about 60% attempts. There were two main parts. Part a) covered three areas linked to user interface design. Part b) covered techniques used to evaluate the usability of a user interface. In part a) it was important to separate design issues of each area and write about the key objectives, independent of type of UI (whether command line or GUI) and underlying framework (web or desktop). Most candidates concentrated on a GUI. Whether GUI or command line some candidates did not address the importance of structure (hierarchical top down navigation) and controls such as menu lists, menu bars or prompts. Within part a ii), the key concepts that were often missing were the need to use controls necessary to facilitate data input, including validation and other data capture approaches. Data output also needed to cover controls that facilitated the consumption and output of data such as reports and graphics. Candidates expressed how the output of information is managed to prevent harm and bias. However some candidates related all parts to providing assistance to some users who find access to computers difficult due to impairment or old age. Part b) was poorly answered. The question asked candidates to explain what techniques are used to evaluate usability of a user interface. Unfortunately many candidates seemingly misunderstood the question and repeated the same answers of those provided in part a). Many candidates simply prescribed the characteristics of usability. The question required knowledge of how to test the usability and to evaluate/ measure the user experience. Therefore some classical testing techniques formed part of the answer as well as the more subjective assessment of the user experience.
A3	
	This was the least popular question in Section A with only 6% of candidates attempting it. Given the relative simplicity of the code, candidates gained credit from the time taken to understand the specification before implementing in code. The key part to the coding involved using the supplied formula to calculate the distance from the centre of the circle. Part b) - many candidates repeated all the classical testing approaches such as white box and black box testing. But with a little more thought given to the fact that the algorithm/ code only produces an approximation, white box testing would not be much use apart from determining how close the output was close to pi. Black box testing would be used to check that the points are correctly classified as being either outside or inside the circle. A further test would be to measure the effect of an increase in the number of iterations. Part c) - the number of iterations is fundamental to the accuracy of the output. There is also a dependency on the randomness of the numbers generated by the random number function and on the accuracy of the square root operation. The solution uses floating point numbers and calculations using these are limited in the number of values they can represent.

A4	
	This was the most popular question in Section A with almost 90% of candidates attempting it. However given the popularity the standard of the answers could be improved. For part a) most candidates covered the essential techniques that a programmer uses to make their code readable. There is a difference between mainly manual techniques (such as adding comments) and tools often provided by an IDE that can embellish code (such as adding indentation). Therefore on careful reading of the question it was a description of techniques rather than IDE tools. Part b) - mostly satisfactory answers which required a clear distinction between system and user documentation that was often absent. Part c) - this was an open question that produced a large array of answers. Generally answers could have been improved upon and many did not address the context of the question which concerned designing a web site that documented a software product. Most candidates produced very generic answers that could apply to any web site. Overall knowledge was lacking of specific user interface design issues such as those that facilitate ease of navigation and accessibility, as well as securing and protecting sensitive information.
B5	
	This was an unpopular question, but some candidates were able to provide good solutions to all three parts. Most candidates gave the correct answer for Part a) but part b) (which requires candidates to write a function) was less well-answered. Most candidates made a reasonable attempt at part c) but many presented code that would not meet the exact requirements of the specification. This question concerned geometric progression. Part a) proved straightforward for almost all candidates. Part b) which involved writing a function for returning the sum was well answered. Some candidates however, failed to correctly use appropriate syntax for the chosen language. Part c) - this part was generally well answered with most candidates able to write a correct selection program.
B6	
	The question covered algorithms. Candidates seemed to misunderstand what was required and struggled to write code that would iterate through the lists looking for common elements. There seemed to be some difficulty understanding the problem and the algorithm required to solve it. Part b) was concerned with finding common elements in two lists. Many candidates could not provide appropriate pseudocode. Answers showed a degree of confusion on indexing appropriately and incrementing the loop, this appears to be an issue with programming rather than understanding the algorithm. Part c) - very few were able to correctly describe the appropriate Big O notation and only minimal marks were available for a narrative description of why sorting would increase the speed of processing.

B7	
	This question concerned software utilities. Most candidates were able to describe the functions of a compiler and to contrast this with an interpreter. Descriptions of the roles of linker and loader could have been clearer, with some candidates not completing these two parts of the question. Some candidates also merged the idea of a linker with linked lists and described how the latter might be used to build data structures such as binary trees.
B8	
	This question was on program debugging. Most candidates were able to write quite a lot in response to this question but some answers were a little muddled. Part a) was generally well-answered however many candidates were vague on detail, discussing debugging in a generalised way rather than focussing on the precise terms of the question. Candidates understood the idea of Black Box testing and were able to give good answers to this part of the question. Part b) - many candidates chose to explain white box testing when the question asked for the role of black box testing, indicating some uncertainty between the two. Most candidates were able to describe example documentation that would arise from black box testing, but answers tended to be quite basic.
B9	
	The question asked candidates to compare pairs of computing terms. The question was popular and generally well answered. Seemingly some candidates did not understand the term “tuple” but most of the other terms were discussed appropriately. Part i) - there were a number of candidates who made errors such as stating ‘lists’ can have only one data type, or tuples are homogenous. Candidates found difficulty in part ii) of this question, to compare operating systems with application software.
B10	
	This was a reasonably well answered question. Candidates were able to provide example code that would result in an infinite loop but the code for part b) was less well-presented. This question proved a popular choice, with most candidates performing well. For part b) many candidates failed to provide any example code and thus gained minimal marks and for part c) a majority of candidates did not provide a meaningful example of an infinite loop although almost all provided a correct narrative.

B11	
	This was another question where many candidates provided full answers. Sometimes the answers were light on detail but there were also some very thorough answers. In general the answers to part b) were more clearly presented than those for part a), where candidates provided a very high-level discussion. This question on testing and maintenance had a mixed quality of response. Part a) concerned acceptance testing and outcomes associated with it. Many candidates simply noted software development testing as seen throughout the lifecycle and did not specifically focus on it being in the latter stages. Part b) was better answered than part a). Most candidates showed a good appreciation of the various maintenance processes.
B12	
	This question was not a popular choice among candidates. Candidates gave good answers to part a) - showing that they understood how CSV files are used. Performance could have been stronger for both sections of the question. Part b) - a number of candidates did not answer this part or gave a brief description of a program to read through a CSV file. For the minority who did offer a pseudocode solution, some were able to gain high marks for completing an example that read through the file looping until the EOF.