Examiner Report	
Qualification Name	Higher Education Qualification
Qualification Level	Certificate in IT
Date/ Series	April 2024
Module	Software Development

General Comments

Section A overall:
Most questions required candidates to write code.

The overriding impression throughout Section A was the inability to provide code that met requirements set out in the question. The examiners felt that the specification/algorithms were clear and well understood by most candidates. Perhaps with the exception of A4, which was mainly avoided.

Those candidates that could write code tended to excel, gaining high marks overall, some achieving the maximum mark.

The examiners would like to emphasise that a proficient command of programming is only achieved by lots of practice in coding solutions to problems (for example on past papers) in languages like Java, Python, JavaScript etc ideally using the many freely available development frameworks and on-line tutorials.

Question no.	comments
A1	This question had four parts and covered syllabus area 1.3 and 1.4. This was a popular question attracting about two thirds of candidate attempts. Overall it yielded the best performing average mark on Section A. Part a) Most candidates could translate the algorithm into code. Most of the marks were awarded on the checks and calculation part of the algorithm. This required accurate translation of the various conditions and calculations into actual program code (of which Javascript, Java and Python were the most popular). Minor errors in syntax were not penalised however there was a small number of candidates who wrote code that resembled pseudocode which was often a rewrite of the algorithm. Inaccuracy in coding the calculations such as the component wages and omitting the total wages was a common reason why marks were lost in this part of the question.

	Part b) Flowcharts have been present on the syllabus and in exam papers for the last couple of years now, yet less than a half of all candidates could accurately derive one from the given algorithm. There was a lack of familiarity with the symbols and the level of detail required in the flowchart. A minor point worth noting was to feed back the outcome of the check for a valid number of hours back to the start rather than to the end (of the program) to allow the user to try again. Part c) Only a small number of candidates fully answered this part. Many candidates did not attempt it at all, or simply repeated the code in part a). It was very important to provide four distinct functions that mirrored the four steps of the algorithm otherwise marks were lost. It was also important to ensure functions had a return clause so that the main program could work through each function call otherwise marks were lost. Part d) Almost all candidates showed a sufficient understanding of why functions add significant value to coding practice.
Question no.	**comments**
A2	This was the second most popular question in Section A and contained three parts with part a) containing 4 equal subparts. This question covered syllabus areas 3.1. 3.3. 3.4 1.1 Part a) Once again the examiner was concerned by the general lack of coding knowledge especially the inability to apply iteration using a "**for loop"** to access an array and manipulating the contents such as doubling the value and filtering out positive numbers. Although the question didn't ask candidates to supply sample data for the array ,it was good practice to do this so that the examiner could verify whether the code worked. Part b) The structure and operation of Stacks and Queues were familiar to most candidates. Examples of their use was often omitted. The best examples related to computing rather than daily human non-computing metaphors. Example a Stack is essential in Recursion. A recursive function makes use of a call stack." The push method appends an item to the top of the stack, while the pop method removes and returns the top item. Part c) Again most candidates were familiar with binary search also called a binary chop. A example that walked through the process gained the highest marks. Some candidates wrote code for a binary search. This was not required only a description of the algorithm with an example.
Question no.	**comments**
A3	This question was a fairly popular question answered by just under half of candidates. It covered syllabus areas 4.1 3.3. However the overall performance on this question was very

	disappointing in comparison to questions A1 and A2. In general answers were shallow and revealed a lack of knowledge of coding, csv files and abstract data types. Part a) was a coding question that many candidates didn't attempt or omitted the code that stored the answer to a file. Part b) Sequential vs Random access was generally familiar but many answers lacked detail and would have benefited from example code that demonstrated an understanding of the differences in accessing a file using random access (say hashing) or sequentially (using iteration). Part c) Again most candidates provided fairly superficial answers mainly concerned with the flexibility of csv files but omitting discussion of the structure and processing of data in a csv file during export/import from apps. The best answers were from candidates who had used csv files in practice often to populate databases using spreadsheets or vice versa. Part d) This part was also answered poorly by many candidates. Some fairly superficial knowledge of ADT was apparent with many candidates unfamiliar with abstract classes in OOP. Answers needed to acknowledge that abstract classes are based on the concept of abstract datatypes. Abstraction has the same meaning in both imperative programming and OOP which involves hiding low-level details with a simpler higher-level concepts whether data or classes. However an abstract datatype is not necessarily an OOP concept. An abstract class is different in that it provides generalisation in OOP by providing a base class for inheritance.
Question no.	**comments**
A4	This was a very unpopular question with only around 10% of candidates attempts. The overall performance was poor only slightly better that A3. Part a) Nearly all candidates didn't know where to start in devising the logic for an algorithm to process Roman numerals. This resulted in very low marks and possibly explains why most candidates didn't attempt this question. Part b) This was the only part where reasonable attempts were made. The main shortcoming was a lack of examples in program code to illustrate how the various functions were applied in practice.

Question no.	comments
B5	This question asked candidates to develop an algorithm and pseudocode to calculate the mean and the mean absolute deviation for a set of numbers in a given data set. The question was extremely unpopular. It seems likely that candidates were daunted by the mathematical nature of the exercise. There were a small number of very good answers from candidates familiar with this type of statistical analysis.
Question no.	**comments**
B6	This was a popular question with many candidates providing good, clear answers. Most candidates made a good job of Part (a) where they were asked to describe four advantages of taking an object-oriented approach to programming. Part (b) was less well-answered with many candidates not understanding about the functional programming paradigm. In their answer to Part (b) some candidates discussed the use of functions in Python or Java rather than in pure functional programming languages like Haskel.
Question no.	**comments**
B7	Another popular question with many candidates gaining high marks. Most candidates were able to describe the file format for extensions *.jpg; *.pdf; *.html; *.csv; and *.txt. to gain a high mark in Part (b).
Question no.	**comments**
B8	This question was generally well-answered. Some candidates listed features of good user interface design such as use of colour and fonts instead of identifying GUI objects such as text fields, drop-down boxes and so on. Some of the answers to Part c) were a little rushed – to gain a top mark candidates needed to think of clear design for a form rather than a rough sketch.
Question no.	**comments**
B9	There were some good answers to this question. Some candidates spent a long-time answering Part (a) which was only worth 2 marks presenting three sets of diagrams where **one** would have been sufficient. The description of arrays and linked lists in Part (b) were generally clear although some candidates did not attempt to describe a linked list.

Question no.	comments
B10	This question was less popular than I would have expected. Relatively few candidates were able to present a flowchart that fully addressed the problem statement but most candidates understood what a flowchart is and were able to use the correct notation.
Question no.	**comments**
B11	This was a popular question. Most candidates were clear about the difference between White Box and Black Box testing but less clear in their discussion of unit testing vs functional testing and scripted testing vs exploratory testing. The answers to Part c) tended to be much shorter than the ones for Parts (a) and (b) with most candidates saying very little about exploratory testing.
Question no.	**comments**
B12	Another popular question. As with the previous question, candidates were generally able to give a full answer to Part (a) discussing compilers vs Interpreters but were less clear about the role of linkers vs loaders and parsers vs lexical analysers.