BCS Higher Education Qualification

Certificate

Date March 2020

EXAMINERS' REPORT

Software Development

General comments
The average mark obtained by candidates at this sitting of the paper was comparable to the highest marks obtained in recent sittings. The pass rate for the exam at this sitting was higher than it has been in recent years.

Question number: A1
Syllabus area: 1.1, 3.3
Total marks allocated: 30
Examiners' Guidance Notes
This was an unpopular question, being attempted by only 12% of the cohort, but most of the students who attempted it demonstrated an understanding of arrays and were able to write clear algorithms. a) The question asked for a suitable data structure to hold these values: **Region** / **Product 1** / **Product 2** / **Product 3** / **Product 4** 1 / 189 / 28 / 132 / 12 2 / 123 / 89 / 356 / 18 3 / 356 / 54 / 128 / 28 4 / 104 / 13 / 108 / 35 5 / 205 / 18 / 102 / 23 Most candidates were able to provide code declaring a suitable array structure. Some confused data structure design with database design and provided a table definition. b) c) and d) For each of these candidates were asked to provide code to process the data. Parts b) and c) were generally well-answered but many candidates did not attempt the more challenging part d). Most candidates were able to write clear code using comments and meaningful variable names to help with this.

Question number: A2
Syllabus area: 1.3. 3.1, 7.1
Total marks allocated: 30
Examiners' Guidance Notes

This question proved to be very popular being attempted by 83% of the cohort. Many students made a very good job of all parts of this question and obtained a high mark. The average mark awarded for this question was 22/30.

a) Asked for code to be presented clearly so that the structure and logic could be clearly understood. Most candidates were able to use indentation to clearly show the flow of control through the code.

b) This part asked for an explanation of the role that identifiers, operators and expressions had to play in the functionality of the code. Most candidates were able to identify these within the code – only a few provided a commentary on the role each component played in the execution of the algorithm.

c) This question asked for a restructuring of the code introducing a WHILE loop to replace the FOR loop in the original. Again, many candidates were able to do this and provided clear answers.

Question number: A3

Syllabus area: 1.3, 2.3, 7.1

Total marks allocated: 30

Examiners' Guidance Notes

This was a moderately popular question being attempted by 45% of the cohort and fairly well answered overall with an average mark of 15/30.

a) Asked candidates to trace the values of variables through the execution of the code. Most people attempting this question were able to make a good job of it. Some candidates presented the trace in an unclear way – something like this would have obtained full marks:

Line 1 .. value of convertedNo = 0;
Line 2 .. message displayed
Line 3 .. n is assigned a value entered by the user – in this case 7.
Line 4 .. dn takes the value 7.
Line 5.. i takes the value 1
Line 6.. starts the loop with j = 7
Line 7.. convertedNo = 1
Line 8.. i = 10
Line 9.. n= 3
Back to Line 6
Line 6.. j = 3
Line 7.. convertedNo = 11
Line 8.. i = 100
Line 9.. n=1
Back to Line 6
Line 6.. j = 1
Line 7.. convertedNo = 111
Line 8.. i = 1000
Line 9.. n = 0
Back to Line 6
Line 6.. j = 0 –exit loop.
Line 11.. Display result

a.

b) Asked for a succinct summary of what the code was trying to achieve. The conversion of denary to binary representations. Most candidates were able to see how this worked.

c) This question required candidates to convert an algorithm into code – again most of the answers to this question were satisfactory.

Question number: A4
Syllabus area: 1.3, 3.4, 7.1
Total marks allocated: 30
Examiners' Guidance Notes
This was an unpopular question being attempted by only 23% of the cohort. All four parts asked candidates to write code to process an array in various ways. The students who fully answered the question generally made a good job of it. Many candidates did not however attempt Part d). The average mark was 13 – this number was depressed by a number of students who chose the question but only attempted Part a).

Question number: B5

Syllabus area: 1.1, 3.3

Total marks allocated: 12

Examiners' Guidance Notes

This was a unpopular question but produced a good performance overall. Candidates were provided with an algorithm that candidates were required to interpret over three parts of each weight of 4 marks.

Part a) required candidates to substitute a given decimal number (69) and show the steps in converting to an octal number. For example:

Step 1 : 69 / 8 => Q8 R 5
Step 2 : store 5 as a character -> "5"
Step 1: 8 / 8 => Q1 R 0
Step 2: store 0 as a character -> "50"
Step 1: 1 / 8 => Q 0 R 1
Step 2: store 1 as a character -> "501"
Step 3: No more divisions possible
Step 4: Reverse("501") => "105"

Although many candidates understood the question, some candidates did not break the processing down with reference to the four steps. Therefore, when asked to "Step through" means exactly that and not to simply write down the answer without any working out.

Part b) An iteration statement is used to repeat a segment of code until either some condition is met or a set number of steps has been completed. In the algorithm an iteration loop is required to read the individual decimal digits (stored in an array) one at a time and then to process the conversion in steps 1 and 2 whilst the loop is running and the condition that the quotient > 0.
This part was answered well overall and most candidates were very familiar with iteration. Examples of iteration statements were welcome but most candidates managed to identify a case for using iteration without any need to write sample code.

Part c) Produced a range of answers for this part.
A LIFO or stack was the most suitable data structure because the order of computation starts by generating the most significant digit which is stored first in sequence and finishes with the least significant digit at the end of the computation. The processing would be more natural and possibly simpler than using an array or using string manipulation.
Most candidates suggested using an array as a data structure. Although an array is not the most suitable data structure it is the most familiar and used. Those candidates who could justify the use of an array were credited with a couple of marks. Some candidates familiar with the Python programming language stated that a built in *reverse* function could return the values in the correct LIFO order.

Question number: B6

Syllabus area: 3.1

Total marks allocated: 12

Examiners' Guidance Notes

This question also had three parts and again proved to be a popular question.

Part a) was worth 4 marks and required candidates to add structure to some supplied code. For example rewritten as

```
if age > 11 then
      if age < 18 then
            print "junior"
      else
            print "adult"
      endif
else
      print "child"
endif
```

Some candidates made the mistake of converting the code into either Java, C or Python often making the code lose its intended nesting and structure in the process. Keeping the code in its original form was expected and should have made it easier to rewrite.

Part b) also worth 3 marks required candidates to express the conditions as English sentences. Some candidates used pseudocode instead of English and hence lost marks. Again a case of not fully reading what the question required.

Part c) also worth 5 marks was overall quite poorly answered with less than half of candidates arriving at the expected answer. Although there are a one or two alternative answers the AND keyword was required to work out the first condition as shown below :

```
if (age > 11) AND (age < 18) then
      print "junior"
elseif age >= 18 then
      print "adult"
else
      print "child"
endif
```

Question number: B7

Syllabus area: 2.4

Total marks allocated: 12

Examiners' Guidance Notes

This was a very popular question and fairly well answered overall with most candidates understanding what was asked.

Part a) was worth 7 marks and required an explanation clearly distinguishing between User and Technical documentation. User documentation describes how to use software and only the essential features of its functionality. It is usually external and printed as a hard copy or provided online for downloading. User documents is intended for non-technical users or with minimal technical terminology and knowledge (with some exceptions) and are written to describe how you use things to achieve an end result. It usually reflects in some of its content what the user has specified in a requirements document recast as a tutorial or walk through. The technical document can do the same but this time show technical detail to show how the functionality was achieved, User documentation should include screen shots to assist in working through an example user interaction. Sources of data and validation checks that apply are often highlighted to alert the user and avoid confusion. At a very basic level, technical documentation is written for the people who manage and maintain the equipment or software, while user documentation is written for the people who use it.

Technical documents are usually written by programmers for programmers with a similar level of technical knowledge, and are more concerned with how things work behind the scenes. Technical documentation includes comments in code for guidance to programmers who may need to maintain; fix bugs or develop programs in the future also included comments inserted into the code by the original programmers. These comments provide guidance or explanations to maintenance programmers who may have to fix bugs in the software.

Part b) was worth 5 marks and related to code readability. It was again fairly well answered.
A naming convention is a set of rules for choosing the character sequence to be used for objects present in code such as variables, constants, functions in source code and documentation. Reasons for using a naming convention (as opposed to allowing programmers to choose any character sequence) is that a well-chosen identifiers make it significantly easier for developers and analysts to understand what the system is doing and how to fix or extend the source code to apply for new needs.
For example,

```
    a = b * c;
```

The above statement is syntactically correct but it's purpose is not evident. Contrast this with:

```
weekly_pay = hours_worked * hourly_pay_rate;
```

Question number: B8
Syllabus area: 2.1, 2.2
Total marks allocated: 12
Examiners' Guidance Notes

This question was very popular with good performance overall. There were two related parts.

Part a) worth 4 marks, required a brief explanation of the contents of a User Requirements Specification (URS). Most candidates could explain that it contains a document that describes the business needs of users and what they require from their system. However many candidates failed to specify at what stage of the SDLC a URS was written (early in the validation process, typically before the system is created) and how the URS are elicited (from the major stakeholders of the proposed software).

Part b) was worth 8 marks with most candidates being familiar with three key techniques (see below) but many candidates did not adequately express the objective of all the various techniques. This should include ways to engage programmers with users in order to collect requirements from various sources, including experienced and new users, SMEs, managers, and, if necessary, the users' customers. Candidates needed to emphasise the contribution of Operational users because they provide some or all requirements for the system's functional and performance capabilities and user interface, by observing the user environment, or capturing their responses to targeted questions. Passive observation is often time well spent.

The key techniques that most candidates focused on were interviews; data from surveys; observation and questionnaires. Other possibilities included Prototyping a version of an application for a user to try out and experiment with. Prototyping is often a good way to get the user engaged and reveal bits of functionality that are not collected from other techniques Users' responses to "is this something like what you want" may open up areas not previously considered. Some of the best answers included a critique of the various methods including benefits and drawbacks.

Question number: B9
Syllabus area: 5.1
Total marks allocated: 12
Examiners' Guidance Notes

This was another popular question and generally well answered with a good range of answers.

Part a) contributed 6 marks Knowledge of dashboards was generally quite poor, or many answers poorly expressed or restricted to web site management tools that call themselves dashboards.

Dashboards are GUI based interfaces that resemble car dashboards and as such are designed to produce information particularly in graphical form (e.g. pie charts) such that it is easy at a glance to assimilate and analyse. E.g. Financial data.

Candidates often stated KPI (key performance indicators) and web-based dashboards without any explanation of why they are used.

Forms resemble paper-based forms because web users fill out the forms using GUI elements/controls checkboxes, radio buttons, or text fields. For example, forms can be used to enter shipping or credit card data to order a product, or can be used to retrieve search results from a search engine.

CLI allows the user to interact directly with the computer by typing commands. Can be used to direct/force the user through a particular route using conditions very much like a flow chart

Part b) contributed 6 marks and was generally answered well by most candidates. Poor answers were from those candidates that did not state which interfaces they were comparing. Most candidates could explain the advantages of CLI over a GUI based Form: Such as having to type very specific commands that computer is able to understand. Not as easy to validate data that has been entered; A CLI being not as intuitive as the equivalent form or dashboard interface and less attractive to end users

Advantages of using command line interface are It is faster than the other types of user interfaces. It is cheaper to use as a lesser resolution screen can be used. Lower resources are used and It doesn't need a windows-based OS to run.

Question number: B10

Syllabus area: 2.5

Total marks allocated: 12

Examiners' Guidance Notes

Part a) Contributed 8 marks roughly 2 marks for each quality factor identified and explained
This was one of the least popular questions on the paper. Overall candidates' performance was below average compared with other questions. Candidates interpreted Quality in many ways with many answers having little regard to the Quality of the software product rather referring to the perceived quality of other factors involved with the process of software development such as testing and design techniques.

The following list show the main factors that contribute to producing a high quality software product that meets certain industry standards. *Candidates needed to include a couple of sentences that describe and justify each of the four factors they selected:*

i. **Portable**: software can run on numerous platforms. Web browsers operating systems
ii. **Usability:** Intuitive and effective user interface easy to learn, or control the system.
iii. **Robust Reliable and Available**
How long the system is up and running and the Mean Time between Failure (MTBF) Minimise the length of time for the system to be down,
iv. **Maintainable**: implies how resilient the code is to change. As a result the terms flexibility and testability cover overall maintainability of a project. Can modifications to the project be done in a timely manner?
v. **Scalable** is the ability for your program to gracefully be scaled up to meet the demand of increased usage. In short, ensuring your program doesn't slow or bust when pounded by more users than you originally anticipated.
vi. **Secure** The software has built in protection to prevent damage caused by malicious use or accident. Safety Operations built into code version control must be present
vii. **Reusable**; can use existing assets in some form within software development; these assets are products and by-products of the software.

Other factors that candidates referred to included version control; technical audit; change management and so on. These would be acceptable factors if there was a clear and concise explanation of how these factors contributed to quality of a software product.

Part b) was worth 4 marks and generally produced good answers covering a range of issues including the following in most cases:

Ethical about re-using existing code, without appropriately acknowledging the source;
Legal if non-disclosure was signed in a contract or when resigning deliberate copying code owned by the company would then be illegal and litigation could be sought.

Intellectual property issues implied by the word discovered implies without any knowledge.

Question number: B11
Syllabus area: 3.4,4.1
Total marks allocated: 12
Examiners' Guidance Notes
This question was the least popular on the paper and also had the worst performance. Candidates showed very poor knowledge of different file organisations such as indexed sequential files and also files often referred to as "flat files" organised in columns and rows or csv files. Each part was worth 4 marks: **Part a)** A Linear search algorithm was the only realistic search algorithm that could be used directly on a file unlike other search algorithms. The algorithm would read the file into memory and then process each item that matches the search criteria. **Part b)** Indexed sequential file though It depends on whether the index is set up on the first field. If it is then the index has the data in this field in sorted order and thus sequential processing can proceed. The use of an index gives more flexibility than a random or sequential file as the search might not start at the beginning of the file. **Part c)** A csv data structure is represented as a type of text file but with a tabular structure (also called a flat file). Each field represents a column and each data item is organised as rows of data items delimited by a comma. Each row is separated by a new line. A csv file is better than an unstructured text file because a programmer would be able to use the delimiters to correctly access individual data items.

Question number: B12
Syllabus area: 2.3
Total marks allocated: 12
Examiners' Guidance Notes
This question was the most popular question on the exam paper but surprisingly overall performance did not match its popularity amongst candidates. Many candidates could not discriminate between a run time error and a syntax error. Many candidates mixed up logical error with a syntax error. Therefore, getting the first part wrong meant the successive parts were usually also wrong. In **Part a)** worth 4 marks a Run time errors cause a program to fail or crash at some stage of its execution. The errors that affect run time behaviour and are not picked up by a compiler (if it's a compiled program) before running the program. When a run time error is found the program stops and an exception handler displays an error message which may not be easy to interpret. Logical errors exist within code that compiles and runs successfully but does not return the expected result due to flaws in the logic of the code. Sometimes logical errors emerge as bugs in commercial code long after a program has been tested and accepted. Thus, more rigorous testing is often required to find the source of these errors. In **Part b**) and **Part c)** worth 2 marks each examples of errors were required. Examples of run time error include: Divide by zero, arrays out of bounds, file not present. The most popular was divide by zero with a line of code to explain the error. Example of a logical error included missing brackets in a calculation so that unintended output was returned because of BODMAS precedence rule: A simple example was expected

Return a * b +2 expected answer (for a = 2 b=3) is 10 whereas the actual result is 8 the brackets should be used to ensure b+2 is computed before the multiplication thus a* (b+2).

Part d) was worth 4 marks and produced a range of answers. Many candidates made the mistake of discussing the manual process of debugging rather than the use of a debugger and the support it provides for debugging a program. As expected, some candidates could describe debuggers, apparently from experience from using a debugger tool in their coding.

A debugging tool checks the code for logical and run time errors as it steps through the code, stopping at a breakpoint before continuing the execution. A breakpoint in the code allows a statement to be checked including values of variables for example.

A debugger also allows code to be checked one line at a time called single stepping allowing the programmer to trace through particular paths of the code pausing at each statement. Thus, the actual paths through the code can be checked against expected paths. Many IDEs (such as NetBeans, Eclipse, Visual Studio) provide extensive and highly interactive aids for debugging code.