BCS Higher Education Qualification

Certificate

May 2021

EXAMINERS' REPORT

SOFTWARE DEVELOPMENT

General comments

In general, candidates performed better in the shorter questions in Section B when compared with the more in-depth questions in Section A. Despite this being a paper on the topic of Software Development, candidates tended to avoid questions that required them to write code.

Questions Report:

Question number: A1
Syllabus area: 1.2, 2.3
Total marks allocated: 30
Examiners' Guidance Notes
The majority of candidates who answered this question were able to provide a good solution to part a). Similarly in part b) the majority of candidates were able to draw a flowchart, however, not all the flowcharts presented corresponded with the solutions presented in part a). There were some cases where candidates who presented a flawed algorithm for part a) obtained good marks for this question because their flowchart represented the logic of their algorithm correctly (meaning that they did not lose further marks for the flaws in the original algorithm). In part c) most candidates were able to make some comparison of the algorithms presented but only a few obtained all the marks available. For full credit to be awarded the following points had to be made: • Candidates should have carried out a dry run of both solutions concluding that they would produce the same results. • Algorithm 2 is more efficient because it increments in steps of two and therefore tests less values. • Although both algorithms produce the same results in this instance, Algorithm 1 tests more values than it should do i.e. 49-51 and 50-52. In some circumstances this could produce incorrect results, for example when the limit is set at 30. Algorithm 2 does not test the pair (2, 4), however this is clearly not a twin prime.

Question number: A2

Syllabus area: 1.3. 2.3. 2.5

Total marks allocated: 30

Examiners' Guidance Notes

The submissions for this question were a little disappointing but many candidates were able to provide at least a partial solution. Something like the following would have attracted full credit:

```
bool is_Letter (char c)
{
    return (c >= 'A' && c <= 'z');
}
bool is_Numeric (char c)
{
    return (c >= '0' && c <= '9');
}
bool valid_Password (char password[])
{
    int charCount = 0;
    int numCount = 0;
    int passLength = strlen(password);
    if(passLength < 10) return false;
    for (int i=0; i < passLength; i++)
        if (is_Letter(password[i]))charCount++;
        else if (is_Numeric(password[i])) numCount++;
        else return false;
    return charCount >= 2 && numCount >= 2;
}
```

Answers to part b) did not depend on a correct solution to part a). Candidates were expected to use white or black box testing techniques to choose their values. The values chosen were expected to check a given feature of the specification. Full credit was awarded for testing a valid password and each of the four conditions specified in part a). No credit was given for testing the same feature twice.

Part c) was generally well answered with most candidates having some awareness of the importance of considering security issues during the software development process. The question was quite open and therefore a wide variety of answers received credit.

Question number: A3
Syllabus area: 2.1, 2.2
Total marks allocated: 30
Examiners' Guidance Notes
This was the second most popular question is Section A of the paper. Candidates were given credit for providing definitions of the three terms in the question but were not given full credit if they did not compare and contrast the styles of programming. In general, candidates gave acceptable definitions of modular and object oriented programming but were less knowledgeable about functional programming. Definitions of the paradigms similar to those found on Wikipedia were acceptable. All three of the paradigms listed in the question concentrate on constructing software by breaking down complex problems into more manageable units. At the same time, they attempt to make these units re-usable so that the time necessary to build additional projects is reduced. In modular programming the decomposition is based on collecting together parts of the code which perform a given task. In object oriented programming tasks are grouped around the data they operate on. Functional programming provides the ability to easily compose complex tasks from simpler ones and guarantees that the composite tasks will behave as expected regardless of the environment in which they execute.

Question number: A4

Syllabus area: 1.1, 2.4

Total marks allocated: 30

Examiners' Guidance Notes

This was the most popular question in Section A of the paper. The majority of candidates who answered it were able to achieve a good mark for the question.

Part a) was straightforward and offered candidates the opportunity to demonstrate an ability to read code and to present it in an easy-to-read form.

A wide range of responses were awarded credit in part b). To obtain full credit candidates needed to both list a technique and give an example of its use. Some candidates lost marks by failing to give examples.

For part c) as in part b) two marks were awarded for each form of documentation accompanied by a brief description of what it entails. Commonly given answers included (but were not limited to):

> Code documentation, usually comments in the code provided to help programmers who have to maintain the code understand how it works.
>
> Test documentation. This will normally accompany test code and should specify what aspect of the program the test is exercising and what the expected outcome should be.
>
> Design documentation. The nature of this will depend on the design technique, It could be pseudocode, a flowchart or UML diagrams. It gives an overview of the design of the software.
>
> Requirements documentation, this will explain what the program is meant to achieve and will set out the basis on which the code can be seen to be performing the function it is required to do.
>
> User guide, this will instruct the use how to use the software, possibly setting out all the screens in the program and the possible interactions.

Again, only partial credit was awarded for naming a type of documentation, full credit was only awarded if a description of the nature of the documentation was also given.

Question number: B5
Syllabus area: 1.3
Total marks allocated: 12
Examiners' Guidance Notes
This question was not popular but was well answered by a majority of candidates who attempted it. Some candidates did simply re state the outlines given in the question and did not attempt to explain or annotate the algorithm, subsequently receiving few marks. In a few cases answers tended to include assumptions about functions that carried out 'black-box' operations that did necessary multiplication and subtraction, in these cases marks were deducted for not fully explaining the algorithm detail. Those candidates who scored low marks clearly could not translate the algorithm into meaningful and precise code. Some of these candidates simply repeated the steps already given and failed to write any meaningful code specifically the required iteration and decision statements.

Question number: B6
Syllabus area: 1.3, 3.4
Total marks allocated: 12
Examiners' Guidance Notes
Part a) of this question was generally well answered and showed an appreciation of the unsorted array given in the question. Some answers indicated an optimised search method using a sorted array spending unnecessary time on answering a question that was not asked. Some candidates misread the question as one of sorting and not searching though it was quite clear what was being asked. In part b), many answers did not follow the requirements of the question and the pseudocode presented did not explicitly mention the place in the array or that the item was not found. Many answers did not present the pseudocode in a coherent way and consequently lost marks

Question number: B7
Syllabus area: 2.2, 3.3. 5.1
Total marks allocated: 12
Examiners' Guidance Notes
This question proved very popular and had a relatively high success rate. For part a) many answers indicated a relatively sparse knowledge of command line properties although most could give a good explanation of the GUI interface properties. Some answers were quite terse single sentences that lacked detail. Part b) did pose some difficulty for many candidates with many unable to articulate a description of a class. Most answers were able to show an adequate understanding of a class and method and did provide key differences. In some cases, marks were lost for not highlighting key differences. Part c) was generally very well answered most candidates did correctly interpret the differences between the data structures. In a few cases, however, answers indicated a misinterpretation between the two structures

Question number: B8
Syllabus area: 2.3, 2.4
Total marks allocated: 12
Examiners' Guidance Notes
Part a) was well answered with a range of explanations emphasising the importance of testing. In part b), most candidates gained some marks for acknowledging test cases but many could not extend the answer to include their importance in unit testing. Very few mentioned the benefits of isolating the tests from other units. Part c), was well answered by almost all candidates. Part d) proved problematic for many candidates Answers tended to miss the point of the question which was to record the execution of a test case. Many candidates tended to answer in terms of software documentation as found in the software lifecycle. A significant number of candidates simply ignored the question. In other cases, a single line answer tended to mention headings without context description. Part d) had the highest mark allocation of this question and ignoring this part severely affected the overall question performance.

Question number: B9
Syllabus area: 2.3, 6.1
Total marks allocated: 12
Examiners' Guidance Notes
This question proved to be difficult for many candidates. Part a) was generally well answered although some candidates found difficulty in going beyond a simple definition and thus failed to gain full credit. Part b) attracted a range of responses to the question. Many candidates found difficulty in citing more than one example in the explanation and tended to explain the use of tools and techniques that were not the subject of this question. Some candidates suggested debugging shouldn't be attempted in an environment with only standard output. Part c) of this question was generally either ignored or partially attempted. Many answers consisted of naming an IDE or selection of development environments taken directly from the question. Very few candidates gave a comprehensive set of IDE's or gave an explanation of how a traditional IDE might be used in debugging.

Question number: B10
Syllabus area: 1.3, 2.1
Total marks allocated: 12
Examiners' Guidance Notes
This question was one of the least popular in Section B of the paper and had a low average mark. For part a), a significant number of candidates could not define recursion. Many answers confused recursion with iteration. Some answers indicated a correct definition of recursion but failed to explain with reference to the example routine that was provided in the question. In part b) many answers tended to repeat the example and failed to articulate the additional requirement of writing an iterative version. A good number of answers mixed the recursive and iterative solution into an unworkable code example. Many failed to gain maximum marks by writing messy and poorly formed code or code that was incorrect in missing parentheses or termination. For answers that used pseudocode, in many cases marks were given for correctly describing the method although maximum marks were rarely achieved due to incomplete descriptions.

Question number: B11
Syllabus area: 3.4
Total marks allocated: 12
Examiners' Guidance Notes
This question was generally well answered. For part a), the majority of candidates could give a reasonable description of two sorting techniques. In some cases, candidates gave a fully comprehensive description of one technique and then gave a sparse description of another two and consequently limited the marks they achieved overall. Many answers were good and gave a thorough breakdown of the steps that the chosen sorting technique would go through. In many cases candidates lost marks by failing to list the differences between the chosen techniques. A fairly significant number of candidates attracted poor marks for incorrectly describing a technique or giving a correct description of the technique but subsequently failing to show correctly how it works. Part b) was correctly answered by many candidates. Some answers were full and comprehensive and gained maximum marks. Those answers which acknowledged time complexity and the correct order notation for a given sorting technique attracted full marks. In some cases, marks were not awarded for answers which wrongly ascribed big O complexity ratings to the technique. In a significant number of cases candidates ignored this part of the question.

Question number: B12
Syllabus area: 1.2, 2.1, 2.3
Total marks allocated: 12
Examiners' Guidance Notes
Only a few candidates were awarded full credit for this question. Although most candidates gained reasonable marks they tended to be unevenly spread across both parts of the question. Many candidates chose to answer only one question from each part. In many cases these answers were very full and comprehensive but restricted the maximum possible marks to half those available for the full question. In a number of cases it seemed that spending a significant time providing an excellent and very full (but unnecessary) answer left the candidate with a shortage of time to fully answer further questions For part a (i) Most were able to describe a flowchart although many had no examples of symbols to aid the description and few adequately described the role of decisions in the flowchart technique. In part a (ii), a minority of candidates offered a description of a mock up. Many ignored this portion of part a) altogether. A number of candidates confused mock up with prototyping in their answers. Part b (i) was well answered by most candidates, the majority were able to describe the essential focus of iterative and incremental techniques. Many did not acknowledge the use of this technique as a way of speeding development and only a few candidates gave the reusability of the prototype as a factor. Part b (ii) This question was very poorly answered. A minority of candidates could give an answer which acknowledged anticipation of failure as a driver of defensive programming. This portion of part b) was ignored by a significant number of candidates. For many candidates the purpose of defensive programming was interpreted as installing anti-virus software and answers focused incorrectly on such packages as the sole purpose of defensive programming.