**BCS Higher Education Qualification**

**Professional Graduate Diploma**

**April 2025**

**EXAMINERS' REPORT**

**Programming Paradigms**

## Questions Report:

| **A1** | |
|---|---|
| | This year's most popular question, attempted by more than 90% of candidates.<br><br>In part (a), most candidates were able to identify differences between procedural and object-oriented paradigms. However, some answers were repetitions of the same difference with slightly different phrasing. Some incorrectly implied that procedural programming languages do not support control structures such as selection and iteration, and that programs in this paradigm are linear. Language was also a little to definite in places where the answer should be more nuanced (for instance, in the procedural paradigm's ability to support encapsulation, etc).<br><br>In part (b), which asked for example software development tasks that are better suited to each paradigm, some answers were a little too abstract, suggesting very broad areas to which one paradigm might be best suited, rather than specific tasks. Other tasks suggested were not well argued (for instance, some candidates argued for implementing a calculator in procedural paradigm, while others suggested object orientation). Whilst the examiner was willing to accept a coherent argument either way, this was not always present.<br><br>Part (c) was not as well answered generally, with some candidates showing a lack of familiarity with the functional paradigm, often confusing it for a language supporting functions in the procedural paradigm. |
| **A2** | |
| | A very popular question, attempted by >85% of candidates.<br><br>In part (a), similarly to question A1, there was a tendency to repeat points with slightly different phrasing, which did not score marks. However, overall, most candidates were able to identify a number of differences between compilation and interpretation. Note that the question asked for differences, not examples of languages, so simply listing languages in each paradigm did not score marks. In some cases, the difficulty in identifying errors in compiled languages was misrepresented.<br><br>In part (b), there was more inconsistency, with some candidates able to identify use cases for each approach, but others making more tenuous suggestions. Most answers centred around rapidity of prototyping and speed of execution, which were valid avenues of reasoning. |

| | |
|---|---|
| | In part (c), which asked about assembly languages, most candidates made valid points around difficult, optimisation/speed of execution, and closeness to hardware. However, a number of candidates did not appear to be familiar with the term and related it to assembly of code in a methodological sense. |
| **A3** | |
| | A very popular question, attempted by >85% of candidates.<br><br>In part (a), following the trend from earlier, there was a tendency to repeat the same point with slightly different phrasing, which did not score additional marks. However, most candidates made thoughtful suggestions concerning the difference between CLI and IDE based programming, mainly focussing on difficulty of use, tool integration, and AI integration.<br><br>In part (b), most candidates made valid suggestions of IDE features that make life easier for the programmer, with the most common answers focussing on syntax highlighting and auto-completion.<br><br>In part (c) there was a split – some candidates clearly knew what step over and step into meant, and some did not, which lead to some creative guesses. |
| **B4** | |
| | An unpopular question, attempted by around 20% of candidates.<br><br>However, more than 75% of those that did attempt the question did well, scoring a pass mark. It could be that more candidates were insecure in their knowledge base in this topic.<br><br>Of those that submitted answers, in part (a), most were able to identify major differences between iterative and imperative approaches. Many also alluded to potential risks in the recursive approach in relation to memory usage.<br><br>In part (b), most were able to offer iterative and recursive implementations of the factorial function, also though some iterative answers had logical errors that would have caused the answer to be miscalculated, so performing a walkthrough is recommended.<br><br>In part (c), which asked for a functional language implementation, some candidates were familiar enough with a functional language to offer an answer (most commonly in Haskell) whilst others were not, and in some cases misinterpreted functional and meaning to implement a function in an imperative language. |
| **B5** | |
| | Like B4, an unpopular question in this year's paper, attempted by around 20% of candidates. However, more than 85% of those that did attempt the question did well, scoring a pass mark, showing secure knowledge of logic languages.<br><br>In part (a), which was only worth 4 marks, some answers were overly long/complicated.<br><br>In part (b), many were able to identify the meaning of existential and universal quantification in predicate logic, although some unduly linked this to logic programming again. Not many candidates convincingly demonstrated the interchangeability of this operators, but instead focussed on their inherent difference in meaning. |

| | |
|---|---|
| | In part (c), most candidates demonstrated facts, rules and queries in Prolog, with some quite elegant answers. Inevitably, there were some syntactic and other errors for which marks were reduced proportionately. Some examples were a little oversimplistic to informatively demonstrate the power of rules and queries. |