

Object-Oriented Programming

The programming exam assesses programming ability against three specific competencies. These cover programming ability in general, plus the elements involved around maintaining legacy software applications and maintaining custom written functionality. In more detail the competencies are:

ICAB5228A – Maintain functionality of legacy code programs, i.e. be able to investigate, understand and enhance software that already exists and is production and was typically developed by other persons. By nature, most legacy software is in the order of years to decades old.

ICAB5230A – Maintain custom software, i.e. be able to investigate, understand and enhance software that may be off-the-shelf or custom-built but in either case has been tailored for the specific situation

ICAB4229A – Apply intermediate programming skills in another language, i.e. be able to develop rudimentary software with minimal specifications provided, in a programming language of choice.

A common scenario is applied across all ACS exam offerings. For the programming exam, a particular business problem with a software solution is identified and then implemented over the course of the questions, and spanning most of the software development life cycle.

In this exam, the situation was to improve the manner in which a recruitment agency stored its schedules of rates, namely, the amounts it pays field employees and that it invoices clients, for the labour services provided. The agency has been using Excel spreadsheets and it is desired to produce a stand-alone application with a back-end database.

Although the competencies assessed require students to code algorithms, the exam does not make any expectations that students can write fully-functional applications, nor perform any complex input or output. Additionally, no specific object-oriented language is stipulated. Most students continue to use Java.

I have to be frank; the results were awful and extremely disappointing. Almost without exception the students could talk at length about how you would gather information, how you might go about testing for faults, but are absolutely lost when it comes to writing a line of meaningful program code.

I considered that the scenario may be difficult to understand with the complex real-world discussion of pay and charge rates and how they are determined – and indeed, many of the government inclusions like superannuation might even be foreign concepts in parts of the world. However, the scenario was not presented solely at the time of the examination. The

training centres have had time to read it and understand it and, importantly, to query it.

Thus, I think the dearth of programming talent is more likely to be caused by insufficient teaching quality within training centres and insufficient opportunities to genuinely program a computer.

Question One

This question told the students they need to locate and understand how the agency presently performs its calculations.

Students were not asked to actually understand or explain how the calculations may work but simply *who* they need to speak to in order to obtain their objective, and *what* questions should they ask. Additionally, *what* documentation should be reviewed?

The response was positive and detailed. The student appeared to be quite familiar with the field of requirements gathering and described employing a rich variety of methods to determine the user's requirements, the documentation of the software and the code and formulas within the spreadsheet itself.

Question Two

This question requested students write programming code that provided a data structure to handle pay and charge rate information. In previous years array handling questions have been mostly well answered. This time around the responses were glaringly deficient.

The question had only basic requirements. Firstly, students were told the data they needed to hold and then asked to define a structure which could hold this information. Perhaps, on reflection, the students needed to be explicitly told how to represent the data, because it is clear few could envision it by themselves.

Fundamentally, the situation called for two structures. The first holds the client ID, the position ID, and will have its own identifying field, eg Rate ID. The second structure then contains fields pay type (eg ordinary time, double time, etc.) pay amount, on-cost amount, margin amount, charge amount, plus the Rate ID to which that rate applies, linking it back to the first structure.

This could be implemented as two arrays, or two objects, or an object with a linked list to hold rates embedded within it or in many other ways.

The responses almost entirely failed to note the comment that at least two objects would be required. Most students tried to make a single valued array, and then to apply a sort algorithm over them – which harks back to a previous question in a previous year. Some students used field names that showed they were similarly trying to answer a question from previous years where a company sought to merge orders for materials to discounts based on economies of scale.

Question Three

Question three was necessary for success; it partially fulfilled all three competencies assessed.

Having considered the data structure required to hold the rates, students were asked to write a routine which displays all the rates in a visually pleasing form.

This question should have been extremely simple. Again, it appears students required far more guidance. Fundamentally, the answer called purely for a routine to iterate through the array or data structure that they had created and merely output fields to the screen. The only part slightly out of the ordinary was that two loops were needed, one working through the client ID and position IDs and the other through the pay types with their individual pay and charge rates for that client/position combination.

There's no doubt in my mind the students were almost universally unprepared for this question but yet it was such a straightforward problem. The answers ranged from non-existent to abysmal with barely a handful of exceptions that would, at a stretch, work even though "a visually pleasing form" had to be given considerable latitude.

The largest problem was the complete failure of students to grasp that they were to assume data existed – indeed, the previous question explicitly stated that no database or file-handling code was required. For the most part the responses here attempted to read lines of input for no clear reason. Any student who wrote code with `BufferedReader` or `ScanLines` or I/O of that ilk were off the track immediately. Some students didn't even display the data they read in thus totally reversing what had been asked. Others echoed back to the user what they had immediately typed in.

This question was extremely poorly answered and every student who failed deserved to fail because their response to this question was completely lacking.

Question Four

The question asked how to locate a bug and how the students may identify and rectify it using extra program code and tools supplied with the integrated development environment (IDE).

The responses indicated an understanding of the different broad categories of faults that could occur and listed several options for troubleshooting these.

Question Five

This question was also mostly well answered. Students were asked how to test their code and introduce it into production.

The response identified a number of test methods and aspects that ought to be considered when putting the code into the working operational system.

Question Six

This question asked what must be considered before deploying new code into production. Specifically, students were asked if there were people they should talk to or documents they should update.

The response indicated several important considerations, and happily, a connection was realised between this and the first question, where documentation had originally been sought. The response received included updating the systems documentation and user manuals they had previously used for reference.