Comments:

Question 1

a) Generally answers failed to identify that key to agile is change and that the smaller the description of an artefact, the easier it is to change. As a model-driven approach seeks to reduce the size of artefact descriptions, it is a technology that matches the needs of an agile process.
b) On the whole, reasonably answered.
c) Answers tended to assume that PIMs and PSMs occur at different modelling levels. However, modelling levels are about the purpose of a model and are a separate dimension to abstraction. Hence, answers failed to identify both PIMs and PSMs as application (M1) level models.
d) Answers generally lacked detail in the two meta models given.
e) Answers generally described two transformations. However, they to compare and contrast these as required by the question.

Question 2

b) This was another question that was, in general, answered very well. The answer I was looking for was that all the story forms except the last were acceptable, since they all describe end-to-end slices of functionality. However, I also accepted answers which pointed out those story forms that were "recommended", in which case only story (i) fits the criteria. As long as the interpretation of the question was consistent throughout the answer, full marks could be awarded for either interpretation of the question.

A number of candidates said in their answers that story (i) did not use the "recommended" format. In fact, the format used by story (i) is also a standard format used by agile developers, and is perfectly acceptable. It is preferred (by some) because it puts the business value (the most important thing) first, rather than last. Even if candidates weren't aware of this alternative format, it does not seem very consistent with the general agile approach to reject a story that contains all the right elements, but just puts them in a different order to the right thing to do, here. But without input from the customer, that is not a decision the team is in a position to make.

A final note on this question: as with part (i), a great many candidates used far too many words in their answers to this question part. Although it was worth just 2 marks in total, a surprising number of candidates wrote from a half to a whole page in answer. Writing 8 or 9 lines of text for a question that is work ½ a mark isn't a very good use of your time in an exam. It is important to check the marks available, and target your answer at the likely amount of information needed.

(That said, I was impressed by the person who attempted a full INVEST analysis of each story form!)

Despite these lengthy answers, there were a worrying number of answers where candidates failed to state clearly (or indeed at all) whether they were saying that the stories were acceptable or not. Where only positive or only negative points were made about a story, I made what seemed to be the sensible inference and marked them as acceptable/not acceptable respectively. But, where a mixture of positive and negative remarks were made, I was unable to award any marks, because the candidate had not clearly stated their opinion one way or the other.

C) There were some very strong analyses of the relative business values of these stories, and some good points made. The main causes of lost marks were:

- Grouping stories together to consider relative values of groups, so that too few points were made about value in the answer as a whole for me to award many marks.
- Answers that focussed only on the technical reasons for the ordering given (e.g. saying that story A would be quicker to implement
once story B was implemented), rather than considering the business value as experienced by the customer.

• Answers that gave business values for all the stories independently, but failed to give any explanation for why they were ordered in the way the customer had chosen. The question was asking about the relative business values of the stories.

• Answers that told the customer he/she was wrong to choose the order they had, based on the technical dependencies between the stories. One of the important benefits of the agile approach is that we are not limited by these old fashioned views of technical dependencies, and can instead choose our implementation order based on business value and technical risk. There is no reason why we can’t implement “Cancel discount” before we implement “Add discount”, if we can find some cost-effective, sensible way to populate the database with discounts to begin with. For example, they may already exist in the customer’s existing systems, or we may be able to import them from another system, as a one-off load process.

D) There were few really good answers to this question, which is not surprising, as it was the part of the question that was intended to distinguish between borderline first class answers and close to 100% answers. Many candidates seemed to misread the question, and so ignored some of the key words within it. The question asked “explain what would be involved in implementing the stories in the order given”. I tried to make this even clearer by including the specific request that answers would “explain how we can write, test and deploy code to display discounts before we have implemented code to create them.” A great many candidates, however, ignored these parts of the question, and instead gave only a vague list of the agile practices they would apply in the development. For example, many people said that they would use TDD, but then said nothing about how using TDD would help the team to implement, test and deploy the stories in the order given. A number of candidates talked about hard coding values, as part of the TDD process, but this is based on a misconception about TDD. We might hard-code values during the TDD process, in order to force us to write better tests; however, we would not allow those hard-coded values to remain in the code for a completed user story (unless they were a genuine constant needed in the domain). We would certainly not allow hard-coded values used in test-writing in TDD to survive into the deployed code.

Some candidates also talked about writing stub code as the solution to this order-implementation problem. This is a way to get started on the implementation and test, but again, we would not normally allow stub code to remain in the deployed code, except in a few very specific circumstances.

E) This question asked candidates to formalise the relative values they came up with in answer to part c) as “proper” story business values. There were some good answers, but many candidates struggled with this question. Many of the supplied business values had not been sufficiently analysed, and needed a few more “why” questions to become good answers. Others were actually functionalities and not business values. Finally, a handful of candidates got very confused about the question and thought it was asking for story size estimates in some form. Needless to say, I was not able to award marks in this latter case.

4.a) Most candidates answered this question perfectly, which was good to see. Unsurprisingly, it was part (iv) that caused the most confusion. This sort of statement could appear in either production code or test code. In the case of test code, it could be used to create the value that will be tested in an assertion, for example. Some candidates were confused about the statement given in part (iii), too, saying that it could be production code, or either production or test code, when really this kind of statement is only likely to be seen in test code. Of course, as some candidates pointed out, we do need to create new instances in production code. However, we would normally be passing in values to the constructor obtained from elsewhere (fields in the GUI, for example, or from an input XML file), rather than having the values hard-coded in the constructor call.

B) This was another question that was answered perfectly by many (though not all) candidates. There was some confusion over part (ii), with some candidates claiming that it must be TFD because the word “suite” implied that multiple test cases existed at the point the developer decides whether to write test code or production code. This is a misconception. Of course, at the very beginning of a piece of TDD, only one test case will exist. But as the coding progresses, the developer should quickly build up a suite consisting of several many tests. As long as the strict red-green-green cycle is adhered to, then TDD is being performed, regardless of the number of tests that exist.

There were other errors made, but as I had only the TAD/TFD/TDD acronyms to go on in most cases, I couldn’t interpret the misconceptions that lay behind them.

C) Again, this was a question that a lot of candidates answered well. There were few perfect marks, however, because while many candidates produced perfect acceptance test table designs, and chose examples that demonstrated sensible scenarios, only a handful of candidates paid attention to the values used in those examples, to make sure they were at the right boundaries. This resulted in a loss of just 1 mark out of 6, however, so there were still many high-scoring answers.

In the low-scoring answers, marks were lost for:
• Omitting key columns from the test (it was common for the weightings columns for the exam/coursework marks to be omitted, and some candidates gave only one “resit” column, so that we could not tell whether it was the exam or the coursework that had been resat).
• Adding unnecessary columns (such as the title of the course unit, or the name of the student). Of course, candidates were not penalised for columns that were needed in order to link a fixture table to the main test table.
• Giving rows that did not match the description of the functionality given in the story (for example, some rows computed the total by just adding the coursework and exam marks together, rather than applying any course specific weightings).
• Having an output column that gave some information about whether a result exam/coursework was needed, rather than computing the total mark for the course unit. (Presumably, these candidates were trying to implement tests based on the business value, rather than the functionality stated in the story.)

d) There were rather more good answers to this question than for the corresponding question in last year’s exam, and quite a few candidates scored perfect marks for their tests. The most common cause of lost marks was for poor design decisions made in translating the tests to objects. For example, several candidates set the coursework and exam marks directly onto the CourseUnit object, meaning that each course unit could have only one exam mark and one course unit mark, regardless of how many students were registered. There was a variety of design errors and infelicities of this kind given in the answers.

A small number of candidates seemed to be unfamiliar with the notion of test code, and wrote some production code to compute the total mark, instead.

E) There were very few really good answers to this part of the question. Many candidates suggested some amalgamation of the
development team and the independent test team (for example, having one member of the ITG work with the developers, or collocating both teams). But only a few marks could be awarded for this point, as (in most cases) few details were given of how this would work, or how it would help to solve the problem. Almost no one considered the important word “independent” in the name of the test team the client wants to bring in. Independent testing is valuable because it can find problems that the development team don’t see, because they are too close to the problem. It is this independence that the client wants to preserve in making their demand. They are not just being stupid by imposing this on an agile team.

In this question, I was hoping that candidates would consider how the benefits of this independence could be combined with the benefits of an agile test-driven approach to development. There were a very small number of candidates who did this in their answers. But, as in the case of question 3 d), this was the part of the question intended to separate out the candidates who are performing at borderline 1st class level from those with a truly excellent understanding of the concepts.