Question 1 (marked by Suzanne Embury)

This question was a popular one (as it normally is on this paper), with around 40 candidates choosing to answer it.

a) There were many reasonable answers to this question, with most candidates revealing a solid grasp of the different kinds of code changes, and the different code reading strategies in their answers. The most common reason for lost marks was simply failing to say enough for me to award 2 marks for the discussion of a single kind of code change. Many answers were correct, but somewhat superficial, making it easy for me to award 1 mark per kind of code change, but difficult to award the full 2 marks.

Apart from this, there were some weaker answers where candidates revealed some significant misunderstandings of the concepts, however. For example, some answers suggested that we always know which method we have to change, when fixing bugs, while others claimed that top-down reading only produces new domain knowledge, and not new technical knowledge of the system.

b) The majority of candidates who answered this question drew a perfect CFG. A few students were confused by my request to draw the graph for only the body of the try-catch block, and added nodes/edges outside the range of lines stated in the question. I neither penalised nor rewarded these answers, marking them only on the nodes and edges for the lines I had asked for in the question.

c) Lots of people answered this question well. The most common mistake was to suggest that the code contained the idiom "Search collection to find first item with some property". The code does not look for the first item with the desired feature, it looks for all the items with the desired property. Because of this, I accepted as a correct answer the idiom "search collection for all items with some property" (which I called "filter" in the notes on Moodle that presented it), since it includes the idea that "all" matching items will be returned as the result and not just the first matching item to be found.

A surprising number of people had trouble giving the language independent statements for the idioms they mentioned correctly. These candidates instead gave a pseudocode description of what this specific code fragment was doing, rather than of the generic behaviour of the idiom. For example, rather than saying "Set current to first item in collection", these candidates said "Find first line in file".

d) Only one candidate gave the answer I was expecting for this question, pointing out that lines 17-19 are implementing a "default value" policy for column names. That is, if a column name is not explicitly specified, this code ensures that it defaults to an automatically created unique name for the table.

Other suggestions also earned marks, although few were convincing enough to earn the full 3 marks available. Suggestions that I accepted as being vaguely plausible include: guarded object creation, guarded addition to a collection, and auto-correction of invalid values.

The answers that earned the fewest marks were those that attempted to cast basic programming operations (such as throwing an exception, adding something to a collection and creating an object) as idioms (rather than individual program statements).

Question 2

a) Generally this part of the question was well answered. A few people lost a mark because they made the same point in different words. For example mentioning both functionality and regulatory changes, which are both forms of requirements change.

b) i. Again generally well answered. There were two main ways in which marks were lost, the first was assuming the decomposability was about the opportunity that the current architecture offered for future restructuring; it is not, it is about the structure of the current architecture. The second way was assuming that decomposability was about separating functionality (e.g. having stovepipes); it is not, it is about the layering used to achieve functionality, e.g. separating UI from service from db access.

ii. Generally this was reasonably well answered. The causes for marks being lost included not including the new design service in the revised architecture, not having an access layer between the service layer and the db, and not justifying the proposed architecture (the question text explicitly asked for this). The major reason for losing marks was a lack of cohesion in the eservice layer, many architectures included a distinct service element for each UI function. In particular, product data spans many of the functions and this needed to be reflected in the architecture.

iii. The answers to this part of the question were the weakest. The question described a scenario and asked for assessment of migration strategies in relation to this scenario. Most answers were totally generic or made only brief reference to the scenario and as a consequence of this lost a significant number of marks. There was no mention/discussion of the order in which functions were migrated and how benefit to the company determined this order.

Question 3 (marked by Suzanne Embury)

This question proved to be a little more popular this year than in previous years, with around 15 candidates choosing to answer it.

A) The most popular answer for this question was the shared database option, in which both companies jointly managed a database into which each contributed data. This was not my own preferred choice (I would have chosen option 3), but I awarded full marks where the choice was sensibly justified, and where (in particular)
candidates had considered the difficulties of jointly managing a database between two companies. Unfortunately, many of the candidates recommending this approach seem not to have thought very carefully about the logistics of it. To take just one example, which company would design the schema for the shared database? Which would correct data quality errors? Which company would take responsibility for administering it? Even presuming that the database was managed on the cloud, where both companies could pay their "share" of the costs, someone with domain and company knowledge is needed to design and maintain the schema, fix problems and authorise access. These issues were ignored by many of those candidates who preferred this option (some of whom seemed to think that the shared database was a copyof "everything" in the databases of the two companies, which made this an even worse option).

Most candidates rightly pointed out the problems with option 1 (allowingeach company limited access to the other's main delivery database), and its advantages. But the analysis of option 3 (each company exports the missed delivery data to the other, so that each company can implement the solution it prefers, without the need to coordinate with the partner company) was weak in almost all cases. Many candidates pointed out the problem that the application building/design work would have to be done twice (once by each company), while not mentioning that the work to be done would be much easier and probably quicker to complete, since agreements on schemas, data semantics, and data administration policies do not need to be reached before progress can be made.

After this, the most common cause of lost marks was simply not saying enough. If only one significant point is made about each option in an answer, then I can't hope to give more than 3 marks for it.

B) There were some good answers for this question, but most people gave the kind of imprecise answers that the question specifically warned against. I was not sure what phrases such as "start of day" or "end of day" would mean in the context of a 24 hour operation such as this. (Of course, deliveries are not made on a 24 hour basis, but the movement of parcels between depots (for example) has to be done overnight so as to maximise the number of parcels that can be delivered during the following day.

Overall, I was hoping for more justification of the specific freshness degree chosen, based on the nature of the business and how this feeds into the decision regarding freshness, in support of the degree of freshness recommended. Some candidates managed this well, but there were few who considered the actual logistics of making the redelivery attempts succeed. For example, several candidates suggested that it would be sufficient if the refresh was completed by the time deliveries had to start the next morning, forgetting that someone has to decide that the redelivery attempt should be made by the other company in time for the parcel in question to be taken to the relevant depot.

C) There were some good answers to this question, but also many answers that earned only 1 or 2 marks. The most common problem was lack of detail. For example, some candidates said that they would use the log to manage the refresh of data, but did not say what it was they proposed to do with the log: what data to extract, by what mechanism and when. I was not looking for a really detailed plan, but I needed more than the statement that a log would be used.

I was also surprised that most people ruled out the possibility of using triggers straight away when the question explicitly said that a couple of triggers could be set up by the team. What could be done cheaply with triggers that would help? Placing a trigger to note the IDs of orders where the delivery attempt has failed that day seems useful and relatively cheap to me…

d) There were some very good answers to this question. Where marks were lost, it was often due to a failure to provide both of the two pieces of information requested for each data quality issue. Almost everyone managed to come up with plausible examples of the data quality issues in the data for this domain. However, a significant number of candidates failed to provide the second answer component requested: an explanation of how the problem might prevent ParcelMe from obtaining value from the collaboration with HomeRun. Some candidates omitted this secod part of the answer altogether, while others talked in very general, high level terms about the consequences, without considering the effect on the value of the collaboration.

A couple of candidates unfortunately gave examples of inconsistency problems, rather than the "representational" consistency requested.

Question 4
a) Generally this part was well answered.
b) Generally reasonably answered. The main points were marks were lost was on the not discussing on-going licencing costs and legal issues related to the quality of building designed using algorithm. Maintenance costs include licencing costs; with bought in components these are unpredictable in the medium to longer term. This factor must be considered when thinking of taking this approach, as is the continued existence of the component's vendor and consequences of them ceasing to support the component. A number of answers mentioned potential legal issues about the software. However, the legal issues are much broader than this. What happens if the building design with the software falls down or injures people because of flaws that can be traced back to the software used to design them?

c) Again, this part was generally reasonably well answered. Minor marks post for similar reasons to the previous part. One notable point was that many answers assumed that an API held stable for at least a year represented a frequent change that could lead to significant maintenance costs. In practice, the potential for an API change every year does not represent a major cost. An API held stable for longer than this will not evolve with current understanding and will result in a need to implement workaround for the incorrect API.