Final Exam – Part 1 - Correction

DysAsso is an association that helps students with learning disabilities such as dyslexia. A member of this association wants to develop a game for smartphone (or tablet) to train students to solve mathematics problems in a ludic way. The game will propose a set of exercises that have to be solved with method cards.
For instance, let's consider the problem "solve the equation $3x^2 – 4x + 5$". Several method cards are proposed to the player: "I want to use the *factoring by inspection* method", "I recognize a *remarkable identity*", "I want to compute *the discriminant*". Once a card chosen, the player is more or less guided to apply it.

DigitalX Company is in charge of the development of the game. The game designer is neither a software engineer nor a mathematics teacher, but he is very available. He has some problem examples that he uses with his children.

1.1 Choose a requirement elicitation method. 1.2 Explain the principles of this method and its limits in this context.	(1,5)

1.1 Several answers can be given. First, an interview can be conducted. An observation phase can be proposed to see how the game designer use his cards with his children. A prototyping phase can also help to check that the requirements are well understood. It is clearly a situation where several iterations in requirement elicitation will be necessary.

1.2 You should give the principles and the limits of the method you choose. Please refer to the lesson.

DysAssociation and the game designer want to propose a first limited version of the game as soon as possible (to evaluate it with their student members). A single category of problem will first be considered (solving an equation) and three method cards. Additional problems, categories and methods cards will be progressively be added.

2.1 Choose a development process appropriate to the situation. 2.2 Explain the principles of this process and justify your choice.	(1,5)

2.1 An iterative method is appropriate.
2.2 The idea is to develop a small subset of the application and deliver it. Then a new part of the application can be developed. Refactoring of the architecture/application might be necesseray.
2.3 I choose this method because it allows to deliver a limited executable version very soon. The game designer will thus be able to provide feedbacks and then to express new requirements.

The game designer would like the game to be executable on all type of systems and easy to use.

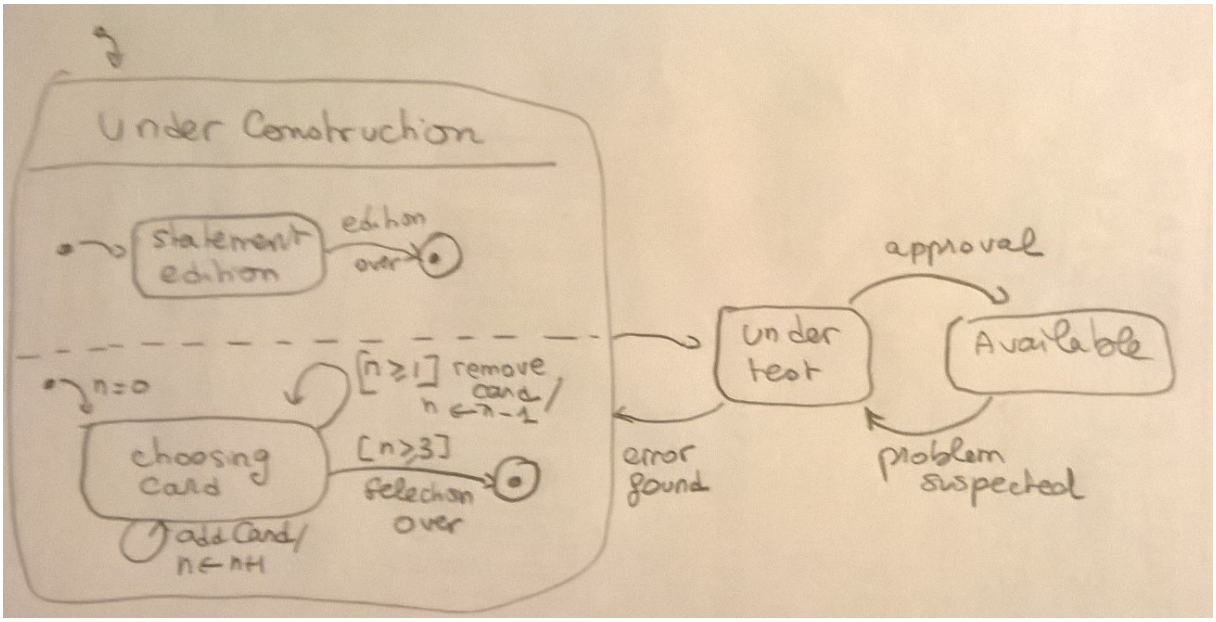
3.1 What is the problem with the formulation of these requirements? 3.2 Suggest a reformulation for each requirement.	(1,5)

3.1 The requirements are not limited nor quantifiable. It will not be possible to check if they are satisfied at the end.
3.2. For the first one, you have to explicit the systems. For instance: "The game will be executable on Windows, MacOS and android."
For the second one, you should precise what elements will be present to help the player. For instance: "On each screen, there is an icon to require help. When one clicks on it, there is an explanation of the actions that are expected from the player". Or "There is an assistance to fill the sheet"…

The design of exercises will follow a life-cycle. First, an exercise has to be written. During this phase, we say that the exercise is "under construction". As soon as the construction is over, the exercise enters in a testing phase. We say that the exercise is "under test". An exercise goes from "under test" to "available" after approval. Some student can report a problem with an exercise when it is available. If it is the case, the exercise returns in testing phase. If an error is identified during the testing phase, the exercise goes back in "under construction" phase.
The construction of an exercise consists of two parallel processes. One of them is dedicated to the problem statement edition. The second is dedicated to the selection of method cards. At least 3 method cards have to be selected before the selection can be declared "over".

4.1 Express the first part of the previous description with an appropriate UML diagram. (4,5) It should include following elements: "addCard", "approval", "available", "choosing card", "editionOver", "errorFound", "problemSuspected", "removeCard", "selection over", "statementEdition" "under construction", "under test"

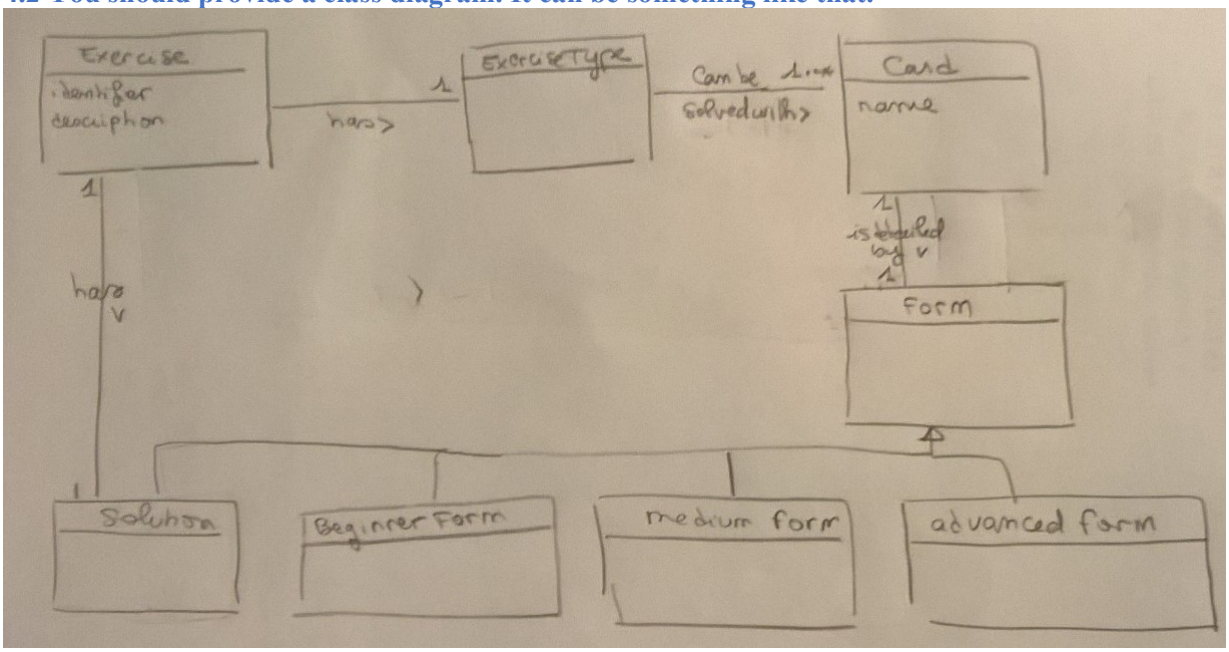
4.1 The first part of the description gives the evolution of the exercise status (or state). You need to use a state diagram. Since the "under construction" stage is composed of 2 parallel processes, you need to have a hierarchical state, with 2 parallel sections.



An exercise has an identifier and a statement description. It has a single type. A type of exercises can be solved with different method cards. A method card has a name. It is associated with a unique form which represent the steps to be followed to apply a method. Reciprocally, a form[1] is associated to a single card. To be able adapt the game to the level of the student (beginner, medium, advanced), we want present the same form with different levels of details. An exercise is also associated to a solution. A solution corresponds to a specific form (it indicates the expected values of the fields).

4.2 Express the second part of the previous description with an appropriate UML diagram.

4.2 You should provide a class diagram. It can be something like that.



[1] A form is a document with fields to fill (*formulaire* en français).

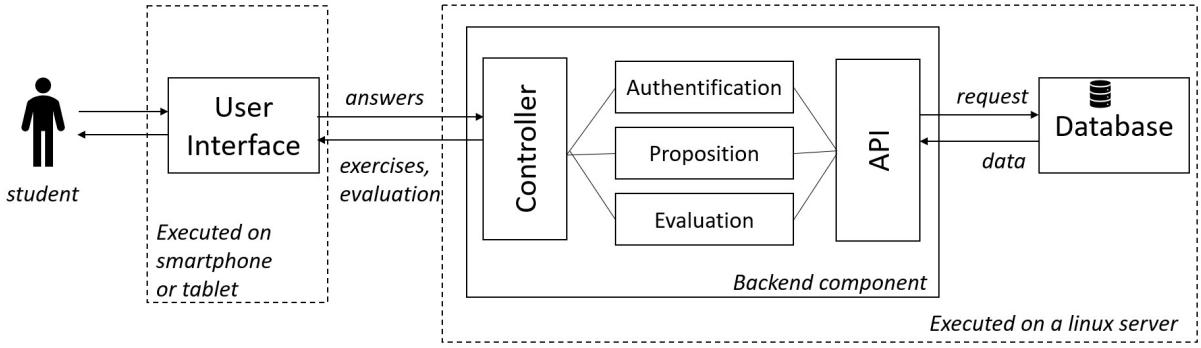
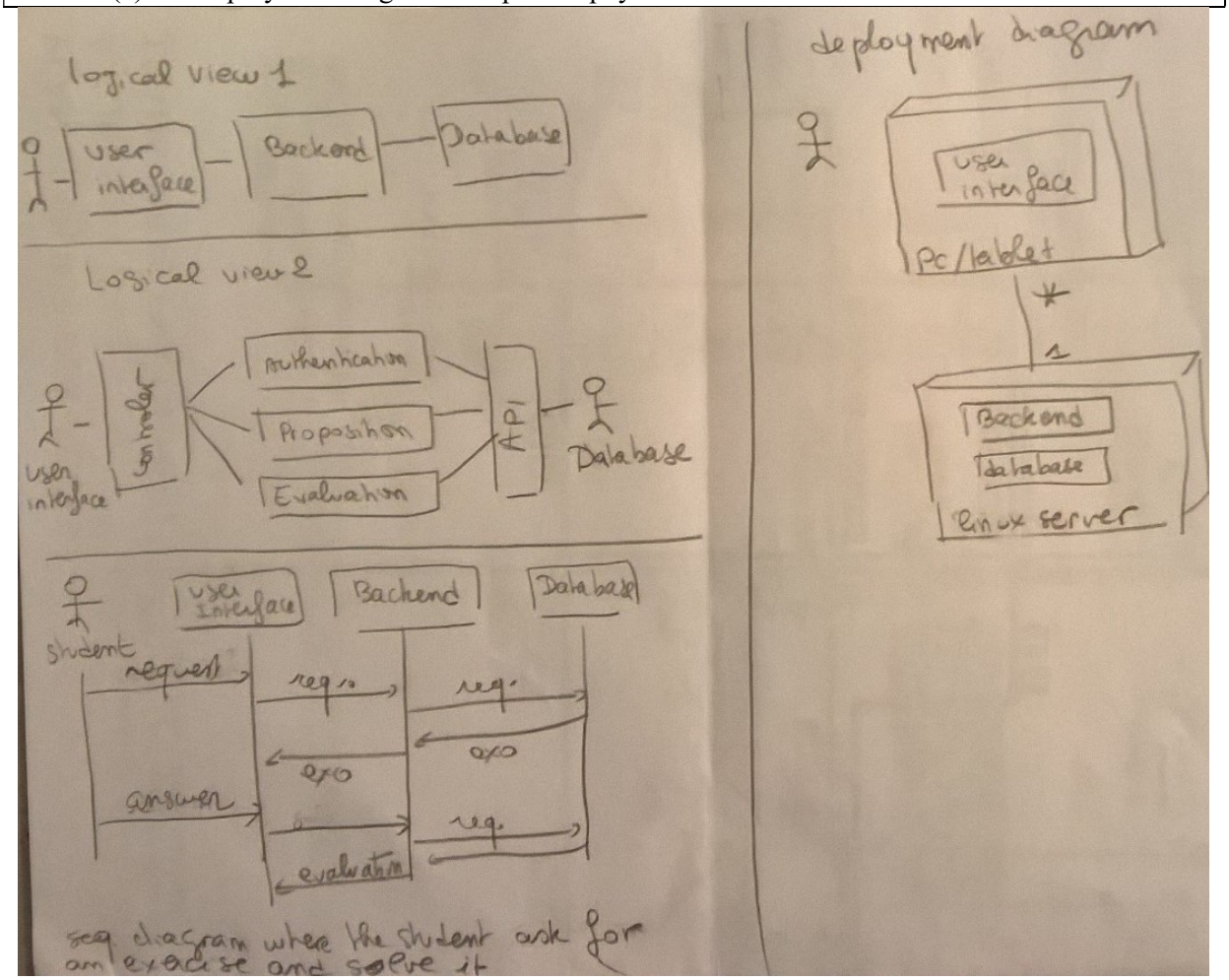


Fig. 1: an architecture

DigitalX Company proposes the architecture given Fig. 1 as a first design.

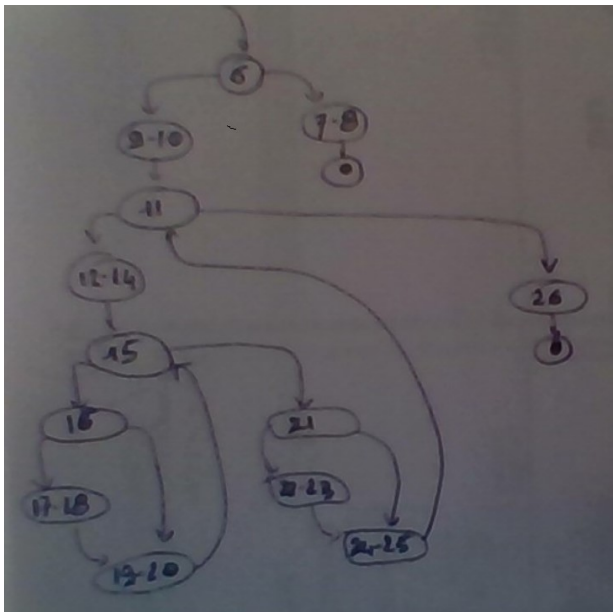
5. Reformulate the architecture proposed Fig. 1 using: (3)
   (a) two component diagrams to express two logical views,
   (b) one sequence diagram to express a dynamical view, and
   (c) one deployment diagram to express a physical view



During the development phase, the function “intersection” is implemented (see Fig. 2). This function returns the intersection of two sets: The result is thus the set of elements that belongs to the two input sets. In mathematics, sets are collections of elements, but here they have been implemented with lists.

6.1 Build the control-flow graph of the intersection function given Fig. 2. (3) 6.2 Give a test set that achieve the statement coverage of the program.

```
5   public ArrayList<Integer> intersection(List<Integer> l1, List<Integer> l2){
6       if (l1 == null || l2 == null){
7           throw new IllegalArgumentException("Lists should not be null!");
8       }
9       ArrayList<Integer> result = new ArrayList<Integer>();
10      int index = 0;
11      while (index<l1.size()){
12          int x = l1.get(index);
13          int i = 0;
14          boolean keep = false;
15          while (i<l2.size()){
16              if (l2.get(i).equals(x)){
17                  keep = true;
18              }
19              i++;
20          }
21          if (keep) {
22              result.add(x);
23          }
24          index++;
25      }
26      return result;
27  }
```



6.2 To cover the states, you need at least a test with a null list, and another in which there is an element in both lists. Since the questions requires “tests”, you should provide the expected results. For instance, you could choose:

intersection(null,[1]) => expected result is an exception

intersection([1],[1]) => expected result is [1]

Note that these tests are very basic and were designed only achieve the state coverage. They were not designed to look for errors in the program. That’s why they are very “poor”.