Test 2017
(https://drive.google.com/drive/folders/1zmCkIqMYYK1JVbTNMjQMvJDOHtCrQmW1?fbclid=IwAR3bisdKphwKpxeyzdFjvs2pNAj1zmnaPfBlHUpXjTa54DlivC1lMogOVPc)

1. When compared to other engineering disciplines, with many, many years, software engineering is really young. Although the term "software engineering" is commonly known today, the very first time it was used with relevance was less than one century ago.
When and Why?

b) The term software engineering was first coined in the 60's, to address the growing importance of software.

In the 60's, software was becoming a very important tool. However, at that time industry santander protocols (or "rules") had not yet been developed since the industry was so young. This led to several problems during the life time of a software, resulting in money lost. The term Software Engineering was first used and defined in the infamous 1968 Nato Conference in response to the problems mentioned above.

2. RUP is heavily supported by UML. In your opinion, in what phase of the process is most UML developed? Elaborate you choice by describing who is responsible for elaborating the UML and what diagrams are most often adopted.

b) Elaboration.
RUP is heavily supported by UML. In my opinion, the elaboration phase is where most UML is developed.
In this stage the developers take a closer look at the project to determine its architecture foundation and to evaluate the architecture in relation to the project.

The UML is the most widely used too to represent all major components, users and their interaction (Model visually). The types of UML's most used to model the architecture visually are:
1. Use-Case Diagram, Class Diagram, State Machine Diagram, Component Diagram, Deployment Diagram: they show a static view of the project.
2. Communication Diagram, Sequence Diagram: they show the dynamic behavior of the project.

The system analyst is usually responsible for the development of the UML, and the project manager contributes/reviews it.

(https://www.webopedia.com/TERM/R/RUP.html)

3. Historically, software development was an inefficient craft. eXtreme programming proposed to change that by introducing novel development practices. What option has practices intrinsic to XP's nature? Briefly describe the identified practices.

c) pair programming, refactoring, test first
Some of the XPs practices are: pair programming, refactoring, test first.

1. Pair programming: two programmers work together at one machine. Driver enters code, while navigator critiques it. Periodically switch roles and pairs. Requires proximity in lab or work environment.
2. Refactoring: improving the structure of the code without changing externally visible behavior. Ex: refactor out any duplicate code generated in a coding session.
3. Test first: Before adding a feature, it's important to write a test for it.
if code has no automated tests, it's assumed it doesn't work. Also, it is important to create tests for bugs discovered before fixing them. This insures that future updates to the code tested don't create bugs.

4. No scrum for us.

5. Requirements engineering refers to the process of defining, documenting and maintaining requirements in the software engineering process. There are different kinds os activities involved, and different types of requirements.

Other important characteristics of the process of requirements engineering are:

c) Requirements engineering is very valued in waterfall-like process, being always the first phase of the development process. More recent processes assume the importance of Requirements engineering along all the process, even at usability studies phases.

Requirements are one of the most important aspects of a project as they describe what the system should (or not) do. Therefore, it is important they are well defined, complete and consistent, hence a discipline dedicated to it. There are different kinds of requirements function and non-functional (as defined on b). However, non-functional might be more critical since if they fail the whole system may not work. Agile projects consider the importance of requirements along all the process and in the modern days, requirements are in constant change and the project and its team must adapt frequently to (sometimes drastic) changes.

6. Software is complex to build. If we were to build all our applications from scratch, software engineering would be an inefficient discipline. Fortunately, software can be composed and reused, enabling us to take on existing software as a scaffold upon which we can build our own.
When building and running the program below in C, which level of software reuse is being applied?

d)
It is object level because we are directly using objects from a library:
we are using the "printf" object directly from the Std library.

7. Software maintenance predication and change prediction are useful because:

c) As software ages and becomes "legacy", if maintaining the system remains feasible and cost-effective.

Maintenance is an importance aspect in software development as maintaining software can be even more expensive than developing new software. Therefore, maintenance and change predication are importance tools to determine if a given software, as it ages, is worth maintaining or should just be totally replaced.

8. Test-driven development promotes smaller, terse and more balanced codebases. What activities/benefits do you think contribute the most to this outcome?

a) Test-first and refracturing.

A Test-driven Development and refracturing are usually related.
They promote a simple tested design.
A test-first methodology helps ensure the developer will create only the necessary code to pass the tests. This way there's a smaller chance the programmer will make unnecessary code and this make a larger codebase.
With this in mind, at each new test, the code addition should be minimal, so refactoring is encourage as a way to, at each step, review what has been done and possibly restructure the code in a better way according to well known refactor techniques, contributing again to a smaller and more balanced codebase.

9. Software project management is concerned with activities to ensure that software is delivered on time, on budget and in accordance with the requirements of the organizations developing and procuring software. One of

the activities of project management is planning the software releases, which features to deliver and when.

Which of the following do you consider the best practice to plan a software project?

b) The developers estimate the time to implement a set of features.

The team reestimates the work effort based on the more detailed assessment and adjusts the features planned for the iteration if necessary.
By doing that, the manager can more accurately estimate the required time for the next release and plan out with the team the feasibility of said features.