Marcel Englmaier
Senior Design XP Exam

1.1 Design, Program, Integration, Production

1.2 The Integration problem presents itself towards the end of the classic life cycle as the classic life cycle does not require constant integration and therefore causes a colossal amount of integration needing to be done.

1.3 XP is very good when it comes to addressing the integration problem. XP states that integration should be done at least once a day with all tests running. This means that the changes that were made to the system are not as substantial as with month-long periods between integration, and the chance to throw a wrench in the works of other developers who then have to integrate with a system that has changed substantially is almost nonexistent, provided the programmers can’t change the system substantially in one day.

2.1 Collective Ownership means that everybody owns the code, and may change any section of the code they wish to at any time.

2.2a Collective Ownership appears when the system is not functioning properly, and it is everyone’s responsibility.
2.2b Collective Ownership appears when a person or a pair sees something they can fix or make better and they do so because they are also responsible for the code. This is mostly seen when sections of code are too complex for their own good.
2.2c Collective Ownership appears all the time as knowledge of the system is spread to everybody because they are permitted, even required, to dig in and explore the system and search for things to work on or upgrade.

2.3a XP addresses the integration problem by requiring daily integrations, limiting the damage an integration can do, and vastly minimizing the time it takes to do an integration
2.3b XP addresses the cost problem of classic projects by focusing on the most important things that are required for function now which also seem to cost less, and relying on the confidence to add costly features tomorrow or in the future when they are required. Sometimes, these costly features become irrelevant or unneeded and in that case, XP hasn’t spent the money on them and is therefore not at a loss.
2.3c XP solves design problems by not attempting to guess what the system will look or be like in the future. Instead, XP focuses on the now and therefore only implements the most important features in a release cycle, allowing the system to change over time. When new features are required, that’s when they’ll be implemented, not when the team thinks a feature might be important in a few months.

3.1 A ‘core’ in XP is a value or principle that XP is founded on.

3.2a Coding
3.2b Testing
3.2c Designing
3.2d Listening
3.2e Pair Programming
3.2f Communication

Marcel Englmaier
Senior Design XP Exam

4.1 Stories in XP are a guideline of functional features that need to be implemented that will be split into individual tasks to complete the story. These are to have enough information to be able to be worked on without the need for another meeting, but not too much as to design the solution. They are a set of requirements for when the story is complete, not instructions on how to do it.

4.2 A good story must have enough information so that the team can work on it without another meeting. A story should be a feature or set of requirements that can be done quickly and split into smaller, easier tasks. But, a story cannot be too big for its own good. A story that is too big and requires too many smaller tasks should be split up into multiple stories to allow for easier work distribution

4.3 Stories are updated by everyone. First, the customer sets some feature requirements. Then the team creates stories for these features. Then, people in the team decide which of these stories they work on. As the people work on the stories, their understanding of the problem and solution will evolve, and can potentially create issues. This is where the customer is then asked to meet to discuss these issues and update or create further stories to solve these problems.

4.4 Because the project will evolve, and a story can become worthless in the future and should be tailored to the current features that are being implemented.

4.5 Classic requirements are much bigger and broader than stories in that classic requirements could be things that the production system must be, and stories focus on current features being implemented. Classic requirements give a picture of the final product, while stories only last a few release cycles at most.

4.6 Stories are bigger and broader than tasks. Tasks are individual things that need to be done to solve or finish the story.

5.1a Version control supports the principle of incremental change, allowing for the system to change slowly and be able to be restored to a previous functioning version
5.1b Unit testing supports the principle of quality work because unit testing does not allow for bad quality to pass through if it is being done properly
5.1c Automation supports the principle of simplicity because automation is designed to do many complex tasks quickly and easily with the press of a button. An example of this would be running 600 unit tests which are able to run through automation. After every change, the programmer can test to make sure the system still works with a simple click of the mouse.