Problem 1

Write a short paragraph to answer these three questions:

- *What are the two major concerns of any software project?*
- *Which of those two do you feel is more important?*
- *Where does the idea of complete functionality fit with these two concerns?*

The two major concerns are delivering working functionality and staying on time and within budget. While both matter, functionality is slightly more important since it is what the customer needs. Complete functionality ties both concerns together by ensuring the product works well and is delivered efficiently.

Problem 2

Write a short paragraph to answer these three questions and briefly explain your opinion:

- *In the Agile method for software development, what are the five main phases that occur in each and every iteration?*
- *Do you feel that any of them could be done at the start of the project and not be repeated in every iteration?*
- *Do you feel that would save time overall on the project?*

The five phases of Agile are gathering requirements, design, coding, testing, and review. These phases should be repeated in each iteration to adapt to changing needs. Skipping any phase might save time at first, but could cause issues later, making it less efficient because larger changes will then be needed later.

Problem 3

Write a short paragraph to answer these four questions and briefly explain your opinion:

- *In the Waterfall method for software development, what are the main phases that occur?*
- *How are they different from the phases in the Agile method?*
- *What other phases are in Waterfall that are left out of Agile?*
- *Do you think these are needed in Waterfall?*
- *Describe a situation using Agile in which one of these extra Waterfall phases might be needed.*

Waterfall has phases like requirements, design, coding, testing, and maintenance, which must be done in order. Unlike Agile, which repeats these steps in short cycles, Waterfall focuses on long, sequential phases. Maintenance is unique to Waterfall and is useful for long-term support. In Agile, a formal maintenance phase might be helpful when managing a complex system. When a project requires extra maintenance to keep the standard functionality running, such as if there are regular variables that must be updated each year, then the maintenance phase is particularly helpful in waterfall since this provides a phase for doing this maintenance, whereas Agile would inherently implies needing to repeat all phases again just to make adjustment to the product.

Problem 4

Write one-sentence answers to the following questions:

- *What is a user story?*
- *What is blueskying?*
- *What are four things that user stories SHOULD do?*
- *What are three things that user stories SHOULD NOT do?*
- *Does the Waterfall method have user stories?*

- A user story is a brief description of a feature from the user's point of view.
- Blueskying is brainstorming freely without limits.
- User stories should: be clear, capture needs, be valuable, and be easy to estimate.
- User stories should not: be vague, too technical, or be overly complex.
- Waterfall doesn't use user stories; it focuses on fixed requirements.

Problem 5

What is your opinion on the following statements, and why do you feel that way:

- *All assumptions are bad, and no assumption is a good assumption.*
- *A big user story estimate is a bad user story estimate.*

We feel that not all assumptions are bad. Assumptions are sometimes necessary, but they should be identified and checked later. Especially when an assumption is part of the story, such as when we assume that existing functionality remains in place. However, there should be caution that assumptions should not be too large in that an incorrect assumption then results in large chunks of functionality that do not work because of incorrect information or needs being assumed.

Large user story estimates are bad because they are harder to predict accurately. By breaking stories into smaller pieces it helps keep things in order and helps to meet the requirements more accurately. Additionally, smaller user stories can also mean quicker delivery to the customer since they can typically be completed quicker than a larger story. This quick delivery helps to build trust and reputation with the customer while also providing the customer with an immediate ROI for their money.

Problem 6

Fill in the blanks in the statements below, using the following things [you can use each thing for more than one statement]: Blueskying; Role playing; Observation; User story; Estimate; Planning poker.

- *You can dress me up as a use case for a formal occasion:* User Story
- *The more of me there are, the clearer things become:* User Story
- *I help you capture EVERYTHING:* Blueskying
- *I help you get more from the customer:* Role Playing
- *In court, I'd be admissible as firsthand evidence:* Observation
- *Some people say I'm arrogant, but really I'm just about confidence:* Estimate
- *Everyone's involved when it comes to me:* Blueskying

NOTE: when you have finished, check your answers with the result in your text on page 62. Do you agree with the book answers? If you disagree with any of them, justify your preferred answer.

Problem 7

Explain what is meant by a better than best-case estimate.

A "better than best-case estimate" assumes things will go perfectly and even better than expected, which is unrealistic in most cases. The better than best case estimate would require that absolutely nothing goes wrong during the development process and furthermore that the initial estimates were wrong so that the project comes in under budget, sooner than expected with more functionality that was intended. This is unlikely to happen because issues can always arise and needs are constantly evolving.

Problem 8

In your opinion, when would be the best time to tell your customer that you will NOT be able to meet her delivery schedule? Why do you feel that is the best time? Do you think that would be a difficult conversation? If so, how could you make it less difficult?

The best time to tell a customer about delays is as soon as you know or even before. When something arises that poses a risk to the delivery schedule, calling it out as a potentially impactful problem creates notice even earlier than early notice of when the problem itself arose. Early notice allows for better planning. While difficult, honesty builds trust, and offering solutions makes the conversation easier.

Problem 9

Write a short paragraph to discuss why you think branching in your software configuration is bad or good, then describe a scenario to support your opinion.

Branching is good because it lets developers work separately, but too much branching can cause problems when merging code. In a scenario where you have a large team, branching can allow a larger volume of work to be completed at once. However, if there is a situation where some of that work may overlap, for example if two developers are working on separate functionality but it relies on a third component that already exists in the product, their changes may impact that pre-existing component and then unbeknownst to the team their branched changes will cause conflict and potential break each other when merged. This is why branching is important and helpful but requires a high amount of communication in order to be effective.

Problem 10

Have you used a build tool in your development? If you have, which tool have you used? What are its good points and bad points — in other words, what do you like about it and/or dislike about it?

Previously used build tools include npm, pip, homebrew, and firebase. The first three are extremely helpful in setting up programs and managing dependencies. However, they abstract a lot of the necessary work from the developer so when there are conflicts or errors it can be time consuming to identify where the issues are coming from. The firebase tools are also very helpful in managing projects through Google Firebase and provide an easy way to build and deploy applications from the command line.