Problem 1

Two major concerns of any software project are how well it works and its efficiency. How well it works means the software does what it is intended to do and meets the requirements. While efficiency pays more attention to how well it performs in terms of speed, memory, and scalability. And between the two, I believe that how well it works is more important than efficiency, because even if it runs quickly and manages memory well, if it is not able to meet the users' needs its useless. Functionality brings the two together because a project can only be considered complete when it both runs well and is efficient.

Problem 2

The agile development method's five main steps are planning, designing, developing, testing, and reviewing. These steps do repeat, so the team working on the project can continuously improve the product and respond to feedback quickly. Some planning and design could be done at the start of the project, but skipping them in future iterations would hurt the flexibility and the quality later down the road. Each of the cycles needs a little bit of the 5 steps to adjust and change to the new requirements. Doing everything once at the start might seem faster, but it would probably lead to more problems down the road when the project needs to be reworked, which would cost more time overall.

Problem 3

The waterfall software development method's main phases are requirements, design, implementation, testing, deployment, and maintenance. Unlike the agile development method, where the steps are repeated in short cycles. The waterfall method follows a set sequence where each phase must be finished before moving on to the next one. Waterfall also a lot more documentation and more upfront planning. While this can cause some extra time at the start, it can be useful for projects that have very strict requirements or have very strict regulations. It would be very helpful in aerospace software, where the extra detailed documentation and review could help save a life.

Problem 4
A user story is a short, simple description written from the perspective of the end user explaining what they want and why.
Blue-skying is brainstorming, which is all about unrestricted thinking and creativity without worrying about restrictions.
User stories should describe the needs, be testable, and be concise, and have a good connection with the developer and the stakeholder.

Three things that a user story should not do are, not include technical implementation details, be too vague, and not fully define their end goal.
The waterfall method does not use user stories, it relies on the detailed documentation defined at the start of a project.

Problem 5
I think that the statement that all assumptions are bad and no assumption is a good assumption is partially true. Assumptions with no clarification can very easily lead to misunderstandings and/or missed requirements if they are not verified. However small assumptions that are tested and communicated can help with progress with the project, and can prevent the project from reaching a standstill.
A big user story estimate is a bad user story estimate
I agree with this statement, because a big user story often means that the project is vague or complicated to be properly estimated. Large storys should probably be broken down into smaller individual stories so it can be easier for teams to track and plan their work.

Problem 6
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:estimate
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 confident: estimate
Everyone's involved when it comes to me: planing poker

Problem 7

A better-than-best-case estimate would be an unrealistic or overly optimistic estimate, where it is assumed there will be no problems in development and that everything will go perfectly and faster.

Problem 8
In my opinion the best time to tell the customer that you will not be able to meet the delivery date is as soon as possible. By telling them as soon as you know, it will help maintain trust between you and the customer. I think that it will be a hard conversation, but being honest and giving them some alternate release dates, it might make it easier

Problem 9
I think that branching in software is good because it allows developers to work on new features or bug fixes without worrying about breaking the main branch. One example could be if

someone needed to add a feature to an already working project. That person could branch it to work on it fully test it then merge back once the feature is working and ready to be implemented.

Problem 10
I have used build tools like npm in some of my classes. I like that they make it easy to run and install stuff. Thepart that i don’t really like is that they sometimes can be hard to set up/fix if not working right. But overall, they have helped me out with projects