Problem 1

The two major concerns of any software project are the questions of how much will the project cost, and how long will the project take. I feel that of the two, the more important one is how long the project will take. If you were to limit the costs of the project, the time that it takes to create the product will not matter because the project will be handicapped through budget restrictions. Monetary restraints are a line that needs to be met for a project to be completed, but time restraints are significantly more dynamic and can affect the project in a lot more ways than money can. Money should not be a concern, if you care about the product then the real important concern will be time restraints, because you can't add more time, but you can give more money. If your goal is to develop the best product that you can then the restraint that you care about is time, not money. Both are an investment, but as time increases the value of the money invested increases, so it is important to not waste time when there is a lot of money involved.

Problem 2

The four main phases that occur in each and every iteration of the Agile method are requirements, design, code, and test. I feel that you could start the project by defining the requirements and design for the project as a whole, and then the developer would be able to refer to the project requirements and design documents when they have questions about what the customer wants. Spending time on these two steps at the beginning of the project can create a roadmap for the developer to follow. I think that if in coding or testing the design needs to change then you will have to repeat the steps, but if there are no major changes then you should be able to reference the initial design. In completing these two steps at the beginning of the development process you will save time later down the road by reducing redundancy in each iteration.

Problem 3

The main phases of the waterfall method that occur are requirements, analysis, design, coding, testing, and operations. This differs from the agile method because each step is designed to a non-repeatable step. Each step will be completed before moving down the waterfall to the next step. The two steps of waterfall that are left out in agile are the analysis and operations steps. I think that the analysis step is just a technical version of the design step, but the operations step is an important step that Agile does not cover. The operations step may be needed in Agile when the customer needs updates to the product once it is released. This could be bug fixing or new feature, but it is an important step that Agile does not account for.

Problem 4

A user story is a description of the product and how a user would interact with the software that is written in the customer's language.

Blueskying is group brainstorming with the goal of capturing all ideas from the group.

A user story should describe one thing the software needs to do, be written using language the customer understands, be written by the customer, and be short.

A user story should not be a long essay, use technical terms, or mention specific technology.

Problem 5

I feel that in general assumptions are bad because you are working in such a technical environment that specificity is everything. There are times where you can make the correct assumption, and time when you need to assume, but in general you should avoid them. Especially because it is so easy to get in contact for a direct answer.
I feel that as long as the user story estimate is honest with what the product is, it is okay to be a large estimate.

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: User story (Wrong initially)
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 (Did not know)
Everyone's involved when it comes to me: Blueskying

Problem 7

A better than best-case scenario estimate is how long the project would take in utopian days where the developer assumes that every single thing that can go right will go wrong.

Problem 8

I think that at the soonest possible time that you are aware that you cannot meet the delivery schedule a meeting with the customer to let them know. It is your job to deliver the product and when you can no longer accomplish your job you need to let the person who hired you know. Sometimes this well let them help you or give you more resources to meet the deadline, but it will also let them plan around the new delivery time. It is not an easy conversation to have because at some level you are admitting failure, but it is the professional and the right thing to do. In a way you know it's the right thing to do because it is the harder thing to do.

Problem 9

I think that branching is good because it lets you explore new paths and ideas without harming the code that is already written. If the branch leads to a new solution then you can follow that branch as the new master path, but if it fails then there is no loss except for time and you hopefully learned something along the way.

Problem 10

I have not used a build tool yet that I am aware of, and looking at them I can see their obvious use, but implementing my own scares me a little. I can see how the option to automate a part of the build process is appealing, but I am not sure if my project will need one. Guess we'll see.