Agile Recap

If stuck, reduce functionality and keep your time, rather than trying to add developers or increase your time/delay. (Might ask on what we think of this and why it is important)
Form own opinion on 'big upfront requirements', consider why some big upfronts, such as planning and designing good architecture, are improtant
Consider the video of the ping-pong ball. First establish a flow, then optimize resources, not the other way round.
Importance of having a relation with your customer. In XP, there are customer roles in the team. What is more realistic is to have some role that interfaces with customers & having customer representatives. Get feedback from customers, but be careful, not all feedback is great - customers don't always know what they want (Nokia customers wanted bigger buttons, then the iPhone came out).
Kano model - WILL BE IN THE EXAM.
Sustainable pace of development, give people around 20% of the time to do what they want, increases productivity
Additive vs multiplicative complexity
Accepting change vs liking change - PMs won't ever like change, but they will accept it
Weighted-shortest-job-first for feature prioritization
Iterative development
Requirements vs scenarios - specifications vs user stories. Requirements cover complete scope of functionalities, user stories cannot replace requirements, they sometimes are needed even if you have user stories.
Scrum ceremonies

- Daily
- Sprint demo
- Sprint retrospective
- Sprint planning
  - Planning poker

Shared code ownership & cross-functional teams (pros and cons)

Pair programming
Test-Driven Development, know the 5 steps in detail, be able to provide examples of the TDD approach. WILL BE IN EXAM
Extreme Programming (Don't need to know practices by heart, but must be able to explain each of them)
Difference between PO, SM, and dev team - Know the principles behind saying that the backlog should be detailed appropriately, and all the other stuff I didn't hear (we should know what a DEEP backlog is)
Difference between project and product Why we want ONE product backlog and not one backlog per project
User stories in INVEST format (be able to explain INVEST)
Be able to break down a functionality into around 10 user stories
Difference between functional and non-functional requirements, their characteristics. Enable stories vs normal stories
What are acceptance criteria, Definition of ready, Definition of Done, differences between these.
Sprints, time-boxed activities. When counting story points, only 100% finished features count
Capacity vs velocity (Capacity is expressed as how much they can do in hours, days, etc., velocity is for (self)comparison, how much has been done in a sprint or space of time, etc.)
Be able to explain a proposed-job allocation as a PO or SM for a team (so meeting time allocations, etc.)
Explain story points, sprint 0
Burnup and Burndown charts. WILL BE IN EXAM
Remember the origins of Lean in manufacturing like 100000000000 years ago or whatever, while scrum is from software
Exaplin main principles of Lean in own words (Reduce waste, minimize WIP - that last one is Kanban but whatever)
Global optimization
Value-Stream mapping - WILL BE IN EXAM

Seeing a development process and answering questions on how we would improve its efficiency
The last responsible moment when making a decision
Queuing theory (Arrival rate, service rate, resource load). Remember the concept of the highway full of vehicles when thinking of resource load
How do small vs large batches pass through this process (why we decompose stories). Draw a diagram showing how batch size affects cycle time.
Difference between Lean and Agile, Kanban and XP, etc. etc.
Understand how Kanban works, the Kanban board, phases. Some examples ask us to show a Scrum board or a Kanban board we would present to our team.
Kaizen, the word for continuous improvement
Know that you can mix Kanban and Scrum - Know which aspects of each can be chosen in some given scenarios. Akin to the exercise we apparently did at home.
Lean focuses on frequent deliveries and quality. XP deals with these indirectly
Agile leadership - The magical triangle of functionality, time, cost but also a fourth vertex in quality for some reason. AUTONOMY, MASTER, PURPOSE, a 'manager' should give these to their teams
Less is more, what is it and how do we interpret it? Don't focus on things that give little value, such as prototyping, focus on the 'low hanging fruit' that offers the most 'instant' value
Deciding what features you would promise to a customer given some tasks, velocities, etc.
Forming, storming, norming, performing, I think
What is the servant leader, difference with a regular leader. Encouraging ideas intstead of giving answers, building trust, people-focused. Leader isn't always expected to give right answers, they should encourage the teams to find their own answers.
Levels of authority, the 7 levels. Don't need to memorize them but be able to explain them or name a few. Which are applicable in given contexts. WILL BE IN EXAM
The video about the submarine captain. Be able to reflect on different leadership styles, what we can gain from each of them. The sphere of influence - don't focus on things you have little impact on.
Be able to reason on the Cynefin framework, even if not known by heart
Different types of reporting, state vs prognosis. How would you report a given status to your manager? WILL BE IN EXAM
Change curve. WILL BE IN EXAM
Be able to answer how commitment level impacts predictability and productivity. Understand why tema should not be overcommiting. WILL BE IN EXAM
5 dysfunctions of a team. Likely will not ask about this but there is a mandatory video on it.
Will ask some random shit about the guest lecture. HOW WOULD YOU WORK/WHAT TOOLS WOULD YOU USE IF YOU WORKED IN A TEAM LIKE THIS? WILL BE IN EXAM
Unit, Integration, System, Acceptance testing. Hardware/Software-in-the-loop testing (what are the challenges of testing HW and HW prototyping → this is about the mandatory reading I think). Complete product testing. Functional vs non-functional tests. TDD. Regression tests & Regression test suite. Daily build. Principles of Contin- uous Integration.
Why is it important to have a single source of truth/information for the software product as well as defined pulling merging procedures
Version vs Revision vs Branch vs Baseline
Test prioritization
Delivery vs Deployment, Continuous Delivery vs Continuous Deployment. Deployment is ensuring that the product is functional and can be downloaded instantly with one click, delivery refers to the actual act of the customer receiving and installing the software.
Deployment Practices (Name at least 3)

- Dark launching
- Feature flag
- Rollouts

DevOps, the conflicts, important that dev and prod environments are not different
sAFE (Scale Agile Frameworks). Showing part of the framework and asking us to explain it, such as showing us the different backlogs, asking how they work, what they contain, which roles are responsible for them, how they are prioritized.
10 sAFE principles, be able to name + describe 3 in own words, or picks 3 and asks us to explain them. e.g. Apply system thinking.
Implementation roadmap. WILL BE IN EXAM (e.g get management support, train people)

Group Manager vs Team Manager
Peer(PF?) Planning (PI Planning as in Product Increment Planning)
Agile Release Train
Our discussion on planning cadence - Develop on a cadence, release on demand
Solution train engineer/solution manager/System architect → know the different roles
Architectural runway/runaway
Conclict between product and solution management (I think it is the conflict between product manager and system architect)
Descentralizing decisions between interfaces
Know about shared services and system team outside of the ART
Understand that there are backlogs and roles on different levels. Epics broken into Capabilities, Capabilities broken into fetures, features into stories
, etc.
Define 'the vision' of a company given its product
Be able to reason why some things are or aren't important given what we have learnt on Volvo cars