# Code Review

* LGTM – Looks Good to Me
* Use a style guide – for everything – coding standard. not only small things, also design/architectural things. Start an empty style guide and fill it as we progress.
* Look at the style guides I started looking at in amazon.
* **Review all CRs within 1 day**.
* **Start high level and work your way down**.
  + Like: re-designing a class interface or splitting up complex functions
  + Save your low-level notes like naming and comments to next iterations
* **Be generous with code examples**. Give them gifts:
  + You might branch the code to create a larger example for the Author like breaking up a large function or adding a unit test
  + Note: reserve this for clear, uncontroversial improvements. If there is argument on if to do this, you will annoy the author and waste your time.
  + Don’t do this more then 1-2 times per review
* Never say ‘you’
  + Can **we** rename this variable to something more descriptive, **like** seconds\_remaining?
  + How about renaming this.… ? what about….?
* **Frame feedback as requests or questions**, not commands
* **Tie notes to principles, not opinions**
  + Instead of saying, “We should split this class into two,” it’s better to say, “Right now, this class is responsible for both downloading the file and parsing it. We should split it up into a downloader class and parsing class per the [single responsibility principle](https://en.wikipedia.org/wiki/Single_responsibility_principle).”
  + Provide supporting evidence where possible in the form of links. They more authoritative, the better!
* Aim to **bring the code up a letter grade or two**. Don’t try taking a D code to A.
  + The only reason to withhold approval is if the code remains at an F (with bugs or so complex that you can’t tell if there are bugs) after a few rounds of review.
* **Limit feedback on repeated patterns**. Instead say – also true throughout the CR.
* **Respect the scope of the review** – only the lines that were changed in the review, unless the CR causes an issue with the previously existing code.
* **Look for opportunities to split up large reviews**: 400 lines is usually the limit. Don’t just ask for a split, offer a logical way to do it (e.g. independent changes etc)
* **Offer sincere praise**
* **Grant approval when remaining fixes are trivial**
* **Handle stalemates proactively**.
  + Talk it out
  + Consider a design review
  + Concede or Escalate.
    - How bad the code really is – what severity issue can it cause?
    - Offer to assign another reviewer or have the manager/tech leader review.
* **Recovering from a stalemate**. Stalemates usually indicate a personal conflict that needs to be resolved.
  + Consult your manager – maybe we’re doing something wrong that causes this conflict
  + Take a break from each other
  + Study conflict resolution

# Project Management

* The Project Management Triangle:  
  You can control two parameters but the 3rd one will be determined by the first two
  + Quality
  + Time
  + Scope
* Discipline Excellence: in the different disciplines: engineering, CI, automation, testing, technology etc.
* Play and endless game (this is not soccer) – plan strategically for the far future (5-years vision) and not tactically. Plan for 5 years and track progress.
* Goal: serve the organisation 10 year forward
* Workers:
  + Smart, hard-working people that create a lot of noise and work – let them work with externals.
  + Smart, smart-working people are the best
  + Stupid and hard-working are the most dangerous because they do a lot of damage.

# Get Buy-In

* Chime-in for Buy-in
* What are you really after?

The best way to get buy-in from stake-holders (or team):

* Relieve a pain-point (be explicit!)
* Be more productive

# Communicate

* Clear communication
* Clear call for action
* Q&A section if relevant

# Scrum

## Post Scrum

This document is for documenting and following up on our action items, items for discussions etc.

### Team Priorities

### Standup

### Tracking

* Sprint Demos (Friday, June 4):
* **Team Learn and be Curious - Team Training List** - please add and vote
* Action Items

### OOTO

### Announcements

### Post Scrum - EVERYONE REQUIRED/Short Ones

### Discussions - You Can Leave (Plz @ people you need otherwise they’ll leave)

# Estimations

1. One developer breaks down the tasks.
2. Used planning poker with additional devs to estimate.

# Questions are the answer

## Project Management

* What is the problem we are trying to solve?
* How do we know it is a problem?
* What does successfully solving this problem look like?
* What solution are we proposing?
  + Is there a better solution – quicker, more efficient, automatic etc?
    - Is there an existing solution we can (re)use?
    - Why aren't we doing that instead?
  + What does this solution cost? In time/people/money.
  + Can we simplify the solution?
  + Are there any parts of the solution that are not required?
* Is this solution permeant or temporary?
  + If temporary - when can we stop doing it?
* How do we track if the solution was (us)successful?
  + What do we do if it’s unsuccessful?
  + What is our contingency plan?
* How do we track this solution?
* Milestones for this solution

## Interrupting Bias

Interrupting Biases:

* How are personal experiences, beliefs or cultural and social norms shaping my perspective?
* Have I sought out new/different perspective before making this decision?
* Did I allow enough time to gather and consider all data versus making the quickest/easiest decisions?
* Am I taking calculated risks or avoiding loss/conflict?

# Interviews

SDE1: coding challenges: will get the same questions as SDE2. Should be able to answer the naive solution.

If they can offer insight to optimised solution (I can improve the algorithm by x) - they raise the bar even if they don’t actually implement it. Can use more guidance.

SDE2: coding challenges: should be able to solve optimised solution with limited guidance.

Before Interview:

\* Make sure that I fully understand the hiring criteria. If the hiring criteria was not given, ask for a pre-brief email or meeting.

Prepare for interview:

\* Request a pre-brief from the hiring manager - what’s the non-negotiable, key LPs etc. Are we covering all the technical aspects of the code? Align on the level? Do we need to consider down-levelling as well?

\* Make sure we have a diverse loop - a good mix of experience interviewers (not just new ones), gender

\* Review the levelling guide and make sure I understand the guide.

\* Make sure you have different interviewers and preferably BR outside of the org in the loop - to make sure we have objective views and there is no pressure from the hiring manager.

\* Read the CV. Understand their environments and how it can influence their LPs.

\* Read notes, previous interviews notes, phone screen etc.

\* We hire for Amazon - if they decide to move teams etc, you should be confident that they can do well.

Writing feedback in this order:

\* Raw notes

\* Competencies:

\* Summary - do it as the end

- my vote, why, highlights, lowlights, seperate functional from LPs. can other interviewers/BR can understand from it why I made this decision?

Interview Stategies:

\* Get the candidate feel as comfortable as possible (not to stress them).

\* Ice breaker: what is your current profile, skills etc that are relevant to the role

\* Concentrate on active listening.

\* When you asking a question, also paste it in the chat. It will help the interview understand and respond better

\* Write prompts in your summary for you and others to anchor to - what do you want to verify etc

\* Frame the question. Be more explicit on what you’re actually looking for

\* You can think about it for a bit. I’m happy to wait

\* If they don’t get your question - frame it a bit differently

Interview Debrief:

\* BR - Facilitator of the conversation in the debrief.

\* Language - don’t use ‘I think’ or ‘I feel’. Lead with the candidate, what they said.

Interview Timelines:

5 min - introduction , water, toilets

15 min - LPs

35 min - Technical question

5 min - questions

Start Interview:

\* make them feel comfortable.

\* Introduce yourself. Explain the process of the interview. We ask a few behavioural questions about leadership principles and then do a coding exercise. We don’t speak to other interviewers until after we submit our report.

\* Questions at the end of the interview - to make sure we cover everything with need.

\* If you don’t understand something - ask. Won’t count against you.

\* I’ll be taking notes on my laptop. I’m listening to you.

\* I may interrupt you. Don’t take offence, I’m just guiding you to answer my specific questions.

\* Toilets/coffee

\* 2 & 5 promise

\* “thanks for offering that information. I can’t and won’t use it to evaluate you so let’s move on.”

\* I might interrupt you to ask you questions. This is not because what you’re saying is not valuable or interesting, it’s just because we have a limited time and I want to get my questions answered so that I can give you the best results.

\* If you are unable to assess one of your assigned competencies due to time constraints, you can request another interviewer cover that area. Be careful not to hint that a competency is a strength or concern prior to the debrief.

\* Questions bank: https://w.amazon.com/bin/view/Amazon\_Interview\_Question\_Bank/ - Note: do the question yourself to see how long/how hard etc.

\* In an interview loop - use the same questions for all interviews. This makes it easier to rank the interviewees.

Troubleshooting:

Tools for verbal communication:

\* Threading - calling back to previous threads in the conversation. It keeps the conversation moving

\* Mirroring - state back what the candidate said in a form of a question -

For candidate that are struggling. We need to get more data without revealing our thoughts.

\* Summarising - read back my notes to the candidate. This sometimes get a candidate a chance to correct the record if we made a mistake.

\* When to move on to the next question? Look at the leadership principle and make sure if you have good data points

\* We can be transparent with the candidate about trying to find examples of how they show specific LPs

LP:

\* Present the STAR - Situation, Task, Action, Result

\* I might interrupt you. Please don’t take offence. I’m trying to guide you toward the information I need.

\* I type. I’m taking notes. I’m listening to you.

\* Sorry just playing back what you told me to make sure I understand.

!! Play back to the interviewee what they said to make sure I understand

Questions:

\* Explain about the next interviewer. Refer him to the hiring manager to ask team-specific questions.

Prepare:

\* 5 questions as backup + follow up questions

\* Good coding questions: a bit vague. See that they can deal with ambiguity

\* Write down expected start time for every section/question to make sure we are tracking.

\* Repeat the course every 2 years.

\* TMAATY - Tell Me About A Time You..

Review:

\* Role of thumb (Shaunak):

\* if LP not strong but no red flag - can be coached as long as technical skills are great.

\* If one low and the other good -> sitting on the fence -> not inclined but open to persuasion.

Interrupting Biases:

\* How are personal experiences, beliefs or cultural and social norms shaping my perspective?

\* Have I sought out new/different perspective before making this decision?

\* Did I allow enough time to gather and consider all data versus making the quickest/easiest decisions?

\* Am I taking calculated risks or avoiding loss/conflict?

SDE1: we expect them to be very good in algorithms and data structures (still fresh from academy)

Phone screens (Sergey L):

—————————

\* Phone screen is used to filter out candidate not to filter in. If someone obviously doesn’t know how to code or raise red flags, we can filter them out

\* If I think that at least one interviewer will be inclined after I let the candidate through, I should let them through.

\* If I’m on the fence, I should usually let the candidate through and calibrate on the on site interview results: if for all the candidates I’m letting through all the interviewers are inclined, I’m probably too strict. If all the interviewers are not inclined, I’m probably too linient.

\* Usually we will not have a second phone screen unless there was an objective reason (emergency etc) why we couldn’t get the data we needed.

\* In the phone screen, probe the candidate if they actually want to work in Amazon

\* phone screen notes are very useful for the onsite interview so write any input you have or things to look at for the onsite.

\* LPs for phone screen: open-ended questions that will discover as many LPs as possible: what’s the most interesting/challenging project etc.

Important: learn and be curious, customer obsession, deliver results and bias for action

\* coding questions: Use questions that can have multiple approaches and that can be easily level up/down if they have problems.

Also: prepare what we want to learn from the coding question. The final solution is not as important as seeing their thought process and how they arrived at the solution.

Ask the candidate what language they are most comfortable with before the coding part.

\* If the candidate is completely stuck - solve the problem with them and guide them.

\* Have a backup question that requires a completely different set of data structures and algorithms

\* If the candidate solves the questions really well and quickly and you have time, try ‘selling them on Amazon’

\* For shadowing/reverse shadowing: don’t share your thoughts until both of you have submitted your feedback.

\* Follow what the candidate is doing: if they have the camera on, turn it on your end as well. If not, use only audio.

\* SDE1: to warm them up to the coding question, ask about data structures relevant for the question (e.g trees etc) and then lead into the actual coding question.

\* Calibration with Seffy - On Design Question:

\*\* L5 should know abs vs interface; when to use inhirit an when to have a field in a class

\*\* L4 might not get it right but if coding is good i am happy

\*\* If coding is mixed but design and basic CS are good i am happy too

\*\* Just give the a few question and look at strengths and if overall we have enough we are incline

Phone Screen - **schedule extra 30 minutes just in case.** In the phone screen, I have to check the followings.

All Phone Screens are calibrated to SDE1 level. Since we don’t have a way to appropriately grudge the level in the phone screen.

\* Coding

\* Prepare when I want to finish each question and hints to get it done in time.

\* Data Structures

\* Problem Solving

\* Algorithms

\* LPs - one question. Don’t spend too much time on this.

\* What’s your favourite language? How confident are you in it? How do you ramp-up your knowledge in this language?

\* Put your fast fail questions at the beginning of the phone interview.

\* Design a Recently Viewed Items Service (scaling/design)

\* Take a problem space you’re working in (GLOW) and distil it to a couple of requirements they can design.

\* If the candidate is super strong - this is a good way to show case some of the cool things we do in Amazon.

—————————————————————————————————————————————————

Calibration Guide for technical skills:

—————————————————————————————————————————————————

\* SDE 1:

\* Deals with Ambiguity:

- Role: Works on defined technology projects. Will occasionally need guidance.

\* Should be able to solve the problem. Maintenance is not that important for SDE 1.

- was the candidate

\* Problem solving - shouldn’t require many hints. But can require some.

\* Data structures and algorithms - did the candidate talked about different data structures and what will be the best one to use?

over-complexity is a risk.

\* Logical and Maintainable code - not a huge concern for SDE 1

\* Did the candidate respond positively to hints? Can they find a solution to a clear problem (not ambiguis)?

\* SDE 2:

\* Deals with Ambiguity:

- Role: Technology strategy is defined. Solution design is not. Delivers independently, but will seek some direction.

\* Logical and maintainable - able to modify the code, clean, naming

\* Problem solving - doesn’t need excessive amount of hints

\* If they don’t understand that they can ask questions: This is the problem statement. Please feel free to ask as many questions as you need.

\* Do SDE 2 ask design questions or just coding? A: Design as well

\* What to do with different background? Example: embedded C programer that wrote code in C and not javascript? String manipulation (create/evaluate math expression)

- A: They'll need to handle it, I can suggest that they can assume a method that return something.

—————————————————————————————————————————————————

LPs:

—————————————————————————————————————————————————

\* If I can’t get the data points, can I ask directly what I’m looking for “Do you know Amazon’s Leadership Principle? Can you give me an example of when you Insisted on the Highest Standard?...”

  \* A: we prefer not to ask direct questions. So, only interject with direct criteria (e.g. example of quality vs speed) if they really require it and mention it in your report.

\* it's not a problem if a candidate reuses a story for more than one question. it might be a problem (an indicator of risk) if a candidate only has 2 or 3 stories for the entire interview. this typically means they lack experience; but it is not a red flag as much as it is a discussion point during the debrief.

\* In loops - maybe we should use the same questions to make sure that we're comparing oranges with oranges.

- A: tell the candidate the properties of the data base. It more about the conversation around solving the question.

The design question should be a conversation. Sophisticated candidate can talk about this and reason about this.

\* Seperate candidate’s current situation (e.g micro-managing boss) from their potential.

## Interview Questions

LPS:

**General:**

**>\*\*Q: What was the most challenging project you worked on?\*\***

\*\*Follow Ups (also for next questions\*\*:

\* What was challenging about it?

\* How did you overcome challenges?

\* What was your specific role?

\* Prioritise?

\* Communicate?

\* Outcome?

\* Lessons learned?

\* Teach/Improve systems?

\* Drill into Something technical?

! After the 1st answer - stop! Do I have what I need? If not, move to the next question!

**## Ice Breaker: - time box!!**

**>\*\* Q: What have you been working on recently?\*\***

\* Find hooks, get chatting. No more than a few minutes.

**## \*\* Troubleshooting \*\***

\* If a candidate doesn’t give me good examples:

- Explain STAR - Situation - Task - Action - Result.

- Remind them to use examples from all their career/life

- If they don’t improve - guide them step by step in the question

- latch on what they talked about and try to drill in. And try extracting data.

- If they are using hypotheticals, remind them I need concrete examples.

\* If the candidate is super upset - suggest that they go for a walk, drink water.

- Explain that we don’t talk between us so they can come back.

- Talk about other things.

- Get the other points and cycle back

\* I need to know exactly what you personally did vs. what the team did.

**## \*\* Insist on the Highest Standards: \*\***

**>\*\* Q: Describe a time when you refused to compromise your standards around quality/customer service, etc. \*\***

**Describe a time where you weren’t happy with the status-quo and worked to change/improve it**

\* Who was the Customer?

\* What was the result?

\* Team? Role?

\* Communicate?

\* Tech/improve system?

\* Lesson Learned? Do differently now?

\* Drill into Something technical/behavioural?

\* Did you ever felt that the quality of the code/project/processes/decissions was not enough?

\* Was there an occasion when the team/manager/customer made the wrong decision?

\* What did you do about it?

\* Did you ever pushed back on their decision and tried to change it?

Criteria:

\*\* minimal/thorough review of others work

\*\* quality vs speed?

\*\* goals - too easy vs challenging but realistic ?

\*\* communicate and gets agreement on expected standards?

\*\* Build systems that are scalable and serve customer needs?

\*\* Continually tries to improve processes?

\*\* Review, feedback, follow up on work/issues?

**>\*\* Backup: Give me an example of a goal you’ve had where you wish you had done better \*\***

\* What was the goal?

\* How could you have improved on it?

\* Team? Role?

\* Customer?

\* Communicate?

\* Outcome?

\* Tech/improve system?

\* Lesson Learned? Do differently now?

\* Drill into Something technical?

\* Was there an occasion when the team/manager/customer made the wrong decision?

\* What did you do about it?

\* Did you ever pushed back on their decision and tried to change it?

Criteria:

\*\* minimal/thorough review of others work

\*\* quality vs speed?

\*\* goals - too easy vs challenging but realistic ?

\*\* communicate and gets agreement on expected standards?

\*\* Build systems that are scalable and serve customer needs?

\*\* Continually tries to improve processes?

\*\* Review, feedback, follow up on work/issues?

**## \*\* Deliver Results: \*\***

**>\*\* Q: Tell me about a time when you had significant, unanticipated obstacles to overcome in achieving a key goal\*\***

\* What was the obstacle?

\* Other options?

\* Outcome? Compare with goal?

\* Lesson Learned? Do differently now?

\* Team? Role? Which part?

\* Prioritise?

\* Trade-offs (quality/timelines/something else)?

\* The right move? Why?

\* Teach/Improve systems?

\* Drill into Something technical?

Criteria:

\*\* Delivered on commitment?

\*\* Enough time&resources for quality?

\*\* Focus on the most important product?

\*\* Multiple projects in parallel?

\*\* Low vs High Quality?

\*\* Excuses vs Persistence?

\*\* Late/Missing Req?

\*\* Communicate?

**>\*\* Backup: Give me an example of a time when you were able to deliver an important project under a tight deadline \*\***

\* What sacrifices did you have to make to meet the deadline?

\* How did they impact the final deliverable?

\* Other options?

\* Outcome? Compare with goal?

\* Team? Role? Which part?

\* Prioritise?

\* Trade-offs (quality/timelines/something else)?

\* Lesson Learned? Do differently now?

\* Teach/Improve systems?

\* Drill into Something technical?

Criteria:

\*\* Delivered on commitment?

\*\* Enough time&resources for quality?

\*\* Focus on the most important product?

\*\* Multiple projects in parallel?

\*\* Low vs High Quality?

\*\* Excuses vs Persistence?

\*\* Late/Missing Req?

\*\* Communicate?

**## \*\* Learn & Be Curious: \*\***

\* seek opportunity to explore new possibilities/prefer to stay with familiar skills

\* Curious about how things work? Learn for expend without immediate deliverable.

\* Search for challenges?

\* Deals with feedback: hide & defensive vs seek & embrace

\* React to negative situations: blame & shame vs learn & grow

\* Self-development?

**>\*\*Q: Tell me about the time you where stuck and did not know what to do next or how to solve a challenging problem \*\***

\* How did you learn what you didn’t know?

\* What are the options you considered?

\* How did you decide the best path forward?

\* What was the outcome?

\* Lesson learned? Do differently now?

\* More details about how you learned all this stuff?

\* Team, Role, which part?

\* Drill into Something technical?

Criteria:

\*\* seek opportunity to explore new possibilities/prefer to stay with familiar skills

\*\* Curious about how things work? Learn for expend without immediate deliverable.

\*\* Search for challenges?

\*\* Deals with feedback: hide & defensive vs seek & embrace

\*\* React to negative situations: blame & shame vs learn & grow

\*\* Self-development?

**>\*\* Backup: Describe a time when you took on work outside of your comfort area \*\***

\* Your initiative/forced to do this?

\* How did you identify what you needed to learn to be successful?

\* How did you go about building expertise to meet your goal?

\* Did you meet your goal?

\* Lesson learned? Do differently now?

\* More details about how you learned all this stuff?

\* Team, Role, which part?

\* Drill into Something technical?

Criteria:

\* seek opportunity to explore new possibilities/prefer to stay with familiar skills

\* Curious about how things work? Learn for expend without immediate deliverable.

\* Search for challenges?

\* Deals with feedback: hide & defensive vs seek & embrace

\* React to negative situations: blame & shame vs learn & grow

\* Self-development?

**>\*\* Backup: Describe a time when someone on your team challenged you to think differently about a problem. \*\***

\* What was the situation?

\* How did you respond?

\* What was the outcome?

\* Lesson learned? What will you do differently now?

\* seek opportunity to explore new possibilities/prefer to stay with familiar skills

\* Curious about how things work? Learn for expend without immediate deliverable.

\* Search for challenges?

\* Deals with feedback: hide & defensive vs seek & embrace

\* React to negative situations: blame & shame vs learn & grow

\* Self-development?

**>\*\* Backup: Tell me about a time when you realised you needed a deeper level of subject matter expertise to do your job well\*\***

\* What did you do about it?

\* What was the outcome?

\* Is there anything you would have done differently?

\* What was the best/worst thing about this experience?

**## \*\* Bias for Action: \*\***

>\*\***Q: Tell me about a time when you have worked against tight deadlines and didn't have the time to consider all options before making a decision\*\***

\*\*Follow Ups\*\*:

\* How much time did you have?

\* What approach did you take?

**## \*\* Earn Trust: \*\***

**>\*\*Q: Tell me about a project where you had to make a change that you anticipated people would have concerns with\*\***

\* How did you communicate it?

\* What did you do to understand the concerns and mitigate them?

\* Were there any changes you made along the way after hearing these concerns?

\* How did you handle questions and/or resistance?

\* Were you able to get people comfortable with the change?

\* What was the result?

\* Lessons learned?

**>\*\*Q: Tell me about a problem you had to solve that required in-depth thought and analysis?** **\*\***

\*\*Follow Ups\*\*:

\* How did you know you were focusing on the right things?

**>\*\*Give me an example of a time when you were able to deliver an important project under a tight deadline\*\***

\* What sacrifices did you have to make to meet the deadline?

\* How did they impact the final deliverable?

\* What was the final outcome?

**>\*\* Q: What was the most interesting project you worked on?\*\***

\* What did you learn?

\* How did you learn it?

\* Did you teach/document it for others?

**## \*\* Deal with Ambiguity: \*\***

**>\*\* Q: Tell me about a project where you didn’t have much guidance or context \*\***

\* Role, which part?

\* How did you find out what needs to be done?

\* What are the options you considered?

\* How did you decide the best path forward?

\* What was the outcome?

\* What did you learn?

\* Lesson learned? Do differently now?

**## CS/OO Basics questions**

==============================

**>\*\*Q: LinkedList, ArrayList, HashMap - Tell me about their characteristics (advantage/disadvantage). When would you use each? What is the complexity of finding an item\*\***

LinkedList, Array, HashMap - Big O (time complexity) of find, remove, add. When would you use each? Which is more efficient for retrieving an element at an index, linked list or array? Why?

**>\*\*Q: Are you familiar with any sorting algorithms? What are they and what are the characteristics? Space/time complexity?\*\***

**>\*\*Q: Design car/ truck/ bicycle/ engine - classes and main attributes only. No need to implement the details\*\***

**>\*\*Q. When would you use an interface instead of an abstract class?\*\***

**## Coding Questions:**

**=================================**

**\* https://w.amazon.com/bin/view/InterviewerBarUnification/SDE/CodingDataStructuresAlgorithms/**

**\* https://w.amazon.com/bin/view/S9/Recruiting/InterviewQuestions/**

\* Linked lists , heaps (min/max) - are not commonly used so not so good for interview questions

\* Clarify requirements?

\* Edge cases?

\* Invalid input?

\* Test code?

\* readable? Functions? Testable?

\* Complexity?

\* solve the problem?

\* Progress pas brute force solution?

\* justify decision? Compare different solution?

\* Consider other solutions?

\* Doesn’t require many hints?

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Coding - Phone screens:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

\* Loops

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Coding - Problem Solving:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

\* Multi-layered question: with naive and optimised solutions.

\* Ambiguous question

\* Ask relevant questions?

\* Solve the problem?

\* optimised solution?

\* Compare different solutions? Justify decisions? Tradeoffs?

**>\*\* Q: Find the products of all combinations of n-1 elements of integers in an n-length array. \*\***

\* Example:

\* Input: [5,6,1,3]

\* Output: [18,15,90,30]

\* Edge cases: empty list, null, one element, element repeating more than once (write the product repeatedly), contain 0 (for devision), max-int,

\* Test code?

\* Complexity?

\* Extension: try not to use division

\* clue: For every index - products left of index.

\* Extension: No devision in better time (O(n))

\* Given a list of meeting time intervals, find the minimum number of meeting rooms required for these meetings.

\*\*\* >\*\* Q: Given availability time for group of people, find the common available times. \*\*

\* Example: Person1= [10-11, 12-14, 15-18](https://hire.amazon.com/), Person2 = [9-11, 12-14, 17-18](https://hire.amazon.com/) returns [10-11, 12-14, 17-18](https://hire.amazon.com/)

\* The data is in 24 hrs format, what if it is 12hr format.

\* If no common times are available, find the time when the max no. of people are available.

\* what if we have more than 2 people?

**>\*\* Backup Q: Write a function that is given two strings representing sentences and returns the words unique to each string.\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Coding - Data Structures and Algorithms:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

\* data structures and algos and I’ll ask a question about cache: first add a cache (expected to use a map), then eviction policy (hopefully LRU)

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Coding - Logical and Maintainable:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

\* Working?

\* Code complexity

\* Readability , naming

\* Functions, reusable, testable

\* Changeable?

\* if you were to make this production level code, would you change anything? - naming, testing etc

\* How would you diagnose issues at runtime?

\* If we have time: write test code

**>\*\* Q. Implement function to convert a string to an int.\*\***

\* good for Logical&Maintainable due to a lot of steps in writing functions and refactoring the code.

\* Example: “1234” -> 1234

\* deal with invalid characters: “12&45”? What do you think is the right thing to do? ok.

\* extension: negative numbers

\* extension: binary. Example: "1001" => 9 , invalid chars: “12”

\* extension: hex. Example: “1a”=>26 , invalid chars: “1h” , valid capitals “1A”

\* extension: any base

\* extension: write a second function that converts eg “abc123xyz677” to [123, 677] . should use his 1st function

\* extension: decimal point

\* extension: negative numbers

\* extension: write a function to evaluate math expression

\* good examples of solutions: https://paste.amazon.com/show/sashri/1613616401

\* **Clarify requirements?** Start with decimal, start with positive only,

\* **Edge cases?** - null/empty string/invalid characters (what do they want to do? Let them decide)/ for hex: low/high case

\* **Test code?** Which test cases will you test? Null, empty, invalid chars, low/high case

\* **readable? Functions? Testable?** Will you change anything before sending this to prod?

\* **Complexity?**

>\*\* Follow up: atof \*\*

**>\*\*Q: Write a method to generate a random maths expression.\*\***

**- good question for SDE 4 , Deal with ambiguity,**

Clarification:

- the method should support the following operators: +, -, \*, /

- the method takes an integer as input, this integer defines how many operators are in the maths expression

- operands and operators are random

e.g. input: 1, output: 2\*52

e.g. input: 3, output: 12/3+231\*1

\* Clarify requirements?

\* Edge cases? 0

\* Invalid input? -1, 0, a million

\* Test code?

\* readable? Functions?

\* Complexity?

\* solve the problem?

\* Progress pas brute force solution?

\* justify decision? Compare different solution?

\* Consider other solutions?

\* Doesn’t require many hints?

\* Removes ambiguity by taking problem statement and further identifying implicit requirements

**>\*\* Follow up: Write a method to evaluate the math expression returned from the previous method.\*\***

Clarification:

- \* and / has same precedence as + and -

e.g. 3+3\*3 returns 18

- Extension: real precedences

\* Clarify requirements?

\* Edge cases? 0

\* Invalid input? -1, 0, a million

\* Test code?

\* readable? Functions?

\* Complexity?

\* solve the problem?

\* Progress pas brute force solution?

\* justify decision? Compare different solution?

\* Consider other solutions?

\* Doesn’t require many hints?

\* Removes ambiguity by taking problem statement and further identifying implicit requirements

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\*\*\*\* Question: \*\* Given two list of integers, return the numbers that appear in both lists**

- good question: both algorithm, loop and using data structures (hash/set) to improve time. Couple this with the question about data types.

— Notes —

Edge cases:

- Duplicated numbers

- Empty list

- One list is big and the other one is small

- Complexity - double loop: O(m\*n)

- Limited memory (sort + walk) - O(n log n) (sort in-place in O(n log n): heapsort) - n log n + m log m + n + m

- Limited time (use set) - O(n) - n + m

- If the lists are sorted

first = [1, 2, 3, 4]

second = [1, 2, 3]

list1 = [1, 3, 3, 2]

list2 = [2, 4, 3]

- SDE 1 should be able to do the naive solution immediately. A strong candidate, the optimised solution (not that hard).

**>\*\* Backup Q: Implement a stack with push, pop. Implement a min function that returns the smallest element in the stack.\*\***

  \* push() pop()

  \* min() [Can be done in O(n) or O(1)]

  \* Complexity of each operation?

\* Clarify requirements?

\* Edge cases? Empty/full stack

\* Invalid input? na

\* Test code?

\* readable? Functions?

\* Complexity?

\* solve the problem?

\* Progress past brute force solution?

\* justify decision? Compare different solution?

\* Consider other solutions?

\* Doesn’t require many hints?

\* Removes ambiguity by taking problem statement and further identifying implicit requirements

Passable Solution:

* min() method simply does a for loop over the existing stack - O(n) min operation, n=O(n) memory

Ideal Solution:

* Two stacks (one to contain the current minimum) - O(1) min operation, 2n=O(n) memory. Can also store the min on each node.

——————————————————————————————————————————————————————————————————————————————————————————

Question

Given an Integer K, and a two dimensional array, write a method that determines whether a duplicate exists in the array given the criteria that the duplicate must be under k distance from its corresponding value.

Input: k=1

Array: [1 0 2 3 ]

 [4 5 6 7 ]

[6 1 3 9 ]

 [3 2 0 1 ]

**Contains no duplicates.**

  Input: k=3

Array:

[1 0 2 3 ]

[4 5 6 7 ]

[6 1 3 9 ]

[3 2 0 1 ]

**Contains duplicates.**

Reference Solution

The following is a reference solution for an efficient answer to the question.

**def** **find\_duplicates**(array, k)

values = {}

array.each\_with\_index **do** |rows, row\_position|

rows.each\_with\_index **do** |\_column, column\_position|

values[array[row\_position][column\_position]] = (values[array[row\_position][column\_position]] || []).concat([{ x: row\_position, y: column\_position }])

**end**

**end**

values.each\_pair **do** |\_value, positions|

**next** **unless** positions.length > 1

positions.combination(2).any? **do** |position\_a, position\_b|

**if** Math.sqrt((position\_a[:x] - position\_b[:x]) \*\* 2 + (position\_a[:y] - position\_b[:y]) \*\* 2) < k

**return** true

**end**

**end**

**end**

**return** false

**end**

——————————————————————————————————————————————————————————————————————————————————————————

**Questions Colin:**

\*\*

**>\*\* Backup Q: Write a function that is given two strings representing sentences and returns the words unique to each string.\*\***

\* Example:

\* simple; your\_func("my dog chased a cat", "your dog chased a cat") => ["my", "your"]

\* repeated words: your\_func("my dog chased my cat", "your dog chased your cat") => ["my", "your"]

\* capitals: your\_func("my Dog chased my cat", "your dog chased your cat") => ["my", "your”, “dog”, “Dog”]?

\* **expansion:** input is a list of strings

\* **Clarify requirements?** Straight forward question. Maybe example?

\* **Edge cases?** - null/empty string(s) / capitalisation? punctuation marks?

\* **Test code?** Which test cases will you test? Null, empty, low/high letters, no unique, multiple in the same string

\* **readable? Functions? Testable?** Will you change anything before sending this to prod?

\* **Complexity?** O(NxM) Can improve? Using set/HashMap

\* Extra expansion: in place (sort + check) - heapsort/quicksort.

\* adequate for sde ii with limited expansion. good for data structures, problem solving and logical & maintainable.

\* meets: arrives at a solution to the problem

\* raises: uses an appropriate data structure to enhance runtime efficiency -- (sde ii) reuses code during expansion

\* sde1: give them boiler-plate code and guide them. SDE 2 - give them vauge verbal description of the question.

—

\*\*

**>\*\* Q: Given a catalogue of products with ranks, write a function that returns the top 10 products.**

expansion 1: instead return the top K products.

expansion 2: the catalogue does not fit in memory (i.e. it is stored as several JSON files).

\* SDE 2 - should find the naive iteration immediately (sort). Extension - using heap/priority queue.

\* Since heap is not widely used, he can prompt them to find another data structure to make them list heap. Won't make them write a heap themselves.

not that good since not a data structure that is commonly used.

See a good solve here: https://hire.amazon.com/interviews/b7e2f327-69a1-4b38-be93-ee91ab2c7b95?type=IN\_HOUSE#/interview\_event

Adrian’s questions:

My phonescreen technical question:

Given a sorted array of ages for all people in Australia, write a function that would return the count of a given target age.

* If they ask about the array, tell them it is sorted (Deals with Ambiguity)
* If this function gets called thousands of times per second, what can you do to make it faster?

[11:08](https://amzn-operations.slack.com/archives/G018CHPU62F/p1612228135053400)

Loop technical question:

Given a front-end web server which logs which writes log files in the format [timestamp, customerId, pageId](https://hire.amazon.com/).

Find the most common page transitions.

FQ: How does this scale? FQ: Many hosts? Each host would pass summarised data to a leader host which would sum for all.

FQ: What about if not enough memory? Can ask each host??

FQ: What if there are multiple machines outputting log files.

**Observations, assumptions and statements.**

1. Confirmed page transition
2. Confirmed that a transition is customer based
3. Confirmed sorting/ording of logs
4. Assumed piped input

srow questions:

Given a sorted array of ages for all people in Australia, write a function that would return the count of a given target age.\*\*

<p>Example: ages = [1, 1, 2, 3, 4, 7, 9, 9, 10, 10, 10, 11....] given target age of 9 should return 2

Seffy’s questions:

**Q: Given sorted list, find the minimum range that contain number from both lists**

**Q: backup - I want to create a game that has a board m\*n every cell can be either a sea 'S' or land 'L'. At the beginning of the game all the cells are sea. Every turn in the game a player can paint a sea cell as a land. Write a class to represent the board game that have a method to place a land cell on the board and returns number of islands that are currently on the board.**.

Sergey L’s questions:

**SDE L5:**

### Q. Design and implement a Least Recently Used (LRU) cache. It should support the operations: get and put.

\* get(key) - Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.

\* put(key, value) - Set or insert the value if the key is not already present. When the cache reached its capacity, it should invalidate the least recently used item before inserting a new item.

**SDE L4:**

### Q. Modify a stack such that there is a method getMinimum() which returns the minimum value in the stack. Only implement standard methods (push, pop, peek) if they require a custom implementation.

https://w.amazon.com/index.php/AIV/CodeInterview/Minimum\_element\_in\_a\_Stack

**Backup question for both:**

### (Backup) Q. You have two numbers represented by linked lists, where each node contains a single digit. The digits are stored in reverse order, such that the 1’s digit is at the head of the list. Write a function that adds the two numbers and returns the sum as a linked list.

345 + 789 = 1134

5 -> 4 -> 3    +    9 -> 8 -> 7    =    4 -> 3 -> 1 -> 1

Shaunak’s questions:

“Given a 2d grid of tiles each with a height of -1, 0, or 1 surrounded by a drain at height 0, determine if a given cell is flooded if it is raining uniformly over the grid”

## Q: Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

# LPs

This insider's guide is meant to bring additional depth of exploration of our Leadership Principles. It's also meant to be owned and maintained by the community at large. Culture is not defined for us -- we define our culture! Please explore our leadership principles by clicking on the "Learn More" links or on the name of the principle itself. These are your Leadership Principles -- read, discuss, and add to the pages linked below. Keep Amazon peculiar!

### [Customer Obsession](https://w.amazon.com/bin/view/LeadershipPrinciples/CustomerObsession)

Leaders start with the customer and work backwards. They work vigorously to earn and keep customer trust. Although leaders pay attention to competitors, they obsess over customers. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/CustomerObsession)

### [Ownership](https://w.amazon.com/bin/view/LeadershipPrinciples/Ownership)

Leaders are owners. They think long term and don’t sacrifice long-term value for short-term results. They act on behalf of the entire company, beyond just their own team. They never say “that’s not my job.” [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/Ownership)

### [Invent and Simplify](https://w.amazon.com/bin/view/LeadershipPrinciples/InventAndSimplify)

Leaders expect and require innovation and invention from their teams and always find ways to simplify. They are externally aware, look for new ideas from everywhere, and are not limited by “not invented here.” As we do new things, we accept that we may be misunderstood for long periods of time. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/InventAndSimplify)

### [Are Right, A Lot](https://w.amazon.com/bin/view/LeadershipPrinciples/AreRightALot)

Leaders are right a lot. They have strong judgment and good instincts. They seek diverse perspectives and work to disconfirm their beliefs. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/AreRightALot)

### [Learn and Be Curious](https://w.amazon.com/bin/view/LeadershipPrinciples/LearnAndBeCurious)

Leaders are never done learning and always seek to improve themselves. They are curious about new possibilities and act to explore them. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/LearnAndBeCurious)

### [Hire and Develop the Best](https://w.amazon.com/bin/view/LeadershipPrinciples/HireAndDevelopTheBest)

Leaders raise the performance bar with every hire and promotion. They recognize exceptional talent, and willingly move them throughout the organization. Leaders develop leaders and take seriously their role in coaching others. We work on behalf of our people to invent mechanisms for development like Career Choice. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/HireAndDevelopTheBest)

### [Insist on the Highest Standards](https://w.amazon.com/bin/view/LeadershipPrinciples/InsistOnTheHighestStandards)

Leaders have relentlessly high standards — many people may think these standards are unreasonably high. Leaders are continually raising the bar and drive their teams to deliver high quality products, services, and processes. Leaders ensure that defects do not get sent down the line and that problems are fixed so they stay fixed. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/InsistOnTheHighestStandards)

### [Think Big](https://w.amazon.com/bin/view/LeadershipPrinciples/ThinkBig)

Thinking small is a self-fulfilling prophecy. Leaders create and communicate a bold direction that inspires results. They think differently and look around corners for ways to serve customers. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/ThinkBig)

### [Bias for Action](https://w.amazon.com/bin/view/LeadershipPrinciples/BiasForAction)

Speed matters in business. Many decisions and actions are reversible and do not need extensive study. We value calculated risk taking. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/BiasForAction)

### [Frugality](https://w.amazon.com/bin/view/LeadershipPrinciples/Frugality)

Accomplish more with less. Constraints breed resourcefulness, self-sufficiency, and invention. There are no extra points for growing headcount, budget size, or fixed expense. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/Frugality)

### [Earn Trust](https://w.amazon.com/bin/view/LeadershipPrinciples/EarnTrust)

Leaders listen attentively, speak candidly, and treat others respectfully. They are vocally self-critical, even when doing so is awkward or embarrassing. Leaders do not believe their or their team’s body odor smells of perfume. They benchmark themselves and their teams against the best. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/EarnTrust)

### [Dive Deep](https://w.amazon.com/bin/view/LeadershipPrinciples/DiveDeep)

Leaders operate at all levels, stay connected to the details, audit frequently, and are skeptical when metrics and anecdote differ. No task is beneath them. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/DiveDeep)

### [Have Backbone; Disagree and Commit](https://w.amazon.com/bin/view/LeadershipPrinciples/HaveBackboneDisagreeAndCommit)

Leaders are obligated to respectfully challenge decisions when they disagree, even when doing so is uncomfortable or exhausting. Leaders have conviction and are tenacious. They do not compromise for the sake of social cohesion. Once a decision is determined, they commit wholly. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/HaveBackboneDisagreeAndCommit)

### [Deliver Results](https://w.amazon.com/bin/view/LeadershipPrinciples/DeliverResults)

Leaders focus on the key inputs for their business and deliver them with the right quality and in a timely fashion. Despite setbacks, they rise to the occasion and never settle. [Learn More](https://w.amazon.com/bin/view/LeadershipPrinciples/DeliverResults)

# Tools

* <https://github.com/in28minutes/spring-master-class/tree/master/01-spring-in-depth>
* https://design-inspector.a2z.com/
* https://drawio.corp.amazon.com/
* https://wsd.aka.amazon.com/
* https://www.jetbrains.com/help/idea/symbols.html
* https://portal.iprd.bluescape.it.a2z.com/organizations/0/workspaces
* https://saymyname.tools.amazon.dev/users/sashri
* https://ion.amazon.com/
* http://jsonlint.aka.amazon.com/
* https://console.harmony.a2z.com/json/
* https://www.epochconverter.com/

# 