# 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.

# Process Checlist

# Process Checklist

## Communications

Excellent examples of communications:

* tickets:
  + My ticket to inform Dexcovery of an upcoming problematic and urgent CR and asking them to help us keep the pipelines open: <https://t.corp.amazon.com/P39653693>

### Before Raising/Flagging Issues:

Time-box and do the followings (1 1.5h pomodoro):

* + What do I want to double check/try/look at before I raise this?
  + Can I offer a solution/workaround/fallback?

## Time Management

* Am I working on the right things? high measurable impact on customer/team
* Reward vs Risk, including lost opportunities. Is the potential of gain outweigh the worst case scenario?

## Project Management

* For every new ask - what is the required outcome? Be exact and include all the outputs + dates!!
* Over communication is better than under communication!
* Send weekly status update
* Anything that might effect the timeline - chime Ray about it to make sure he sees and act on it.
* Be explicit about my consideration and time - why did I decide to do this? what else is taking my time?

## Requirements

* What is the issue we’re trying to solve? customer perspective! Get a full understanding of the requirements!
* Test! Thoroughly! ALL platforms and ALL scenarios!!
* Get the data - metrics etc.
* Most requirements are optional - see how hard it is. Find alternatives and get back to the business. Maybe we don’t really need this. Maybe the priorities are different.
* Start with what the customer needs to see and work backwards from there - which data do we need, where can we get it from etc.

## Design

* Is there a change in the down-steam services we’re calling (which ones/TPS)? If there is - we need to take into account if the service can handle our TPS and take care to onboard/update them on the expected changes.
* Data privacy - are we using/sharing/storing private data that we shouldn’t?
* No latency regression (time-to-interactive)
* milestones
* weblabs
* tests
* Look at the code. Jump in and play if I need better perspective.

## Estimations

Take into account:

* On Calls schedule
* Vacations
* Time to deploy and release, especially on mShop, including release dates for mShop
* Time for OE stuff, setup, refactorings etc. Ask Ray/google - rule of thumb for refactoring etc.

## Metrics

* See customer impact of issue - see severity definitions.

## Weblab

* Create a weblab to activate the new feature
* Add metrics to log weblab’s different treatments (C, T1, T2 etc)
* Add weblab to table
* Verify weblab triggers metrics

## Code

* Created Design (see above)
* Create a weblab to protect the code (see [Weblab](https://quip-amazon.com/4Y3ZAepXVhxh#JSS9CAOdUl1))
  + Make sure that the weblab will be easy to remove (both code and tests should default to T1 with C being the special case).
* Write the code. Create logical commits
  + Validate API inputs, including format and min/max values. Defensive coding!
* Add metrics for everything interesting:
  + Functionality - what we want to know that happened
  + Errors - what will help us debug/recognise errors
  + Weblab - what will help us knowing when a weblab is no longer used and can be wrapped up.
* Add unit tests
* Add integration tests on Beta and Gamma
* Run Coral Tests - make sure I didn’t break anything and that my changes are working
* Run E2E tests + Make sure metrics are working. Important: test on ALL platforms and ALL scenarios!!
* Publish CR(s)
  + Boy Scouts Rule: Unrelated - reduce commit warnings
* Inform QA on feature, weblab and tests

### Code Commit

* + Good commit message - will include the why of the change. The 7 rules of git commit messages (<https://chris.beams.io/posts/git-commit/>):
    - [Separate subject from body with a blank line](https://chris.beams.io/posts/git-commit/#separate)
    - [Limit the subject line to 50 characters](https://chris.beams.io/posts/git-commit/#limit-50)
    - [Capitalize the subject line](https://chris.beams.io/posts/git-commit/#capitalize)
    - [Do not end the subject line with a period](https://chris.beams.io/posts/git-commit/#end)
    - [Use the imperative mood in the subject line](https://chris.beams.io/posts/git-commit/#imperative) (fix c rather then fixed c or fixing c).
    - [Wrap the body at 72 characters](https://chris.beams.io/posts/git-commit/#wrap-72)
    - [Use the body to explain *what* and ***why*** vs. *how*](https://chris.beams.io/posts/git-commit/#why-not-how)
  + and also (<https://dhwthompson.com/2019/my-favourite-git-commit>):
    - Explain the reason for the change
    - It’s searchable
    - It tells a story
    - It spread the knowledge

## CR

### Before Publishing:

* Read my code with fresh eyes: what might confuse me? Review again the next morning and then send to review.
* Write a clear change list description: explain the logic and context of your cahnge. A good changelist description explains **what** the change achieves, at a high level, and **why** you’re making this change.
* Split unrelated changes into separate CRs
* Seperate refactoring from functional changes:
  + Add tests to exercise the existing behavior (if they’re not already there).
  + Refactor the production code while holding the test code constant.
  + Change behavior in the production code and update the tests to match.
* Try to limit the size of the CR.
* Thank and appriciate your reviewer work!
* If your reviewer made a mistake. Think how can you make your code more obviously correct and less confusing?

### While Reviewing: <https://mtlynch.io/human-code-reviews-1/>

* *If I don’t understand the comment, I can use:*
  + *What changes would be helpful?*
* Award all ties to your reviewer - they have a better perspective on how it is to read this code fresh. So unless you have actual objection to their comment, defer to their suggestion
* Once you send your code out, driving the review to completion should be your highest priority. Delays on your end waste time for your reviewer and increase complexity for your whole team.
* Be generous with code examples
* Never say ‘you’
  + **What about renaming***this variable to something more descriptive*
  + *Can***we***rename this variable to something more descriptive, like*seconds\_remaining*?*
* Frame feedback as requests not as commands
* Tie notes to principles, not opinions

### After the Review:

* Are there repeated errors/themes I need to pay attention to?

## Testing

### JS

Run the followings and verify that they **work as expected** and **not log any errors to the console**:

* re-open GLOW multiple times without changing anything
* Change address to zip-code
* Change address to country
* Change address to delivery
* Change address to pickup-store
* Run [Sanity tests](https://quip-amazon.com/aGaOAWKjLyYV/GLOW-Sanity-Testing)
* Test your feature - 2 times

## Bugs

* related to missing weblab?
* Related to dialled-up weblab? See: <https://tiny.amazon.com/1hv45lzwb/mcmamazsear>
* Graphs - what is the impact of this bug?
* Stop it doing more harm
* progress the ticket a bit. Just do enough to improve the situation..
* If it’s not important enough to do now, will it ever be important? If not → close the ticket.

## On Call

Roles (in priorities):

1. first responder - LSE, Auto cuts, tickets
   1. Is there a customer impact? If not, lower priority
   2. If this is a manual cut - trust the requester but verify the impact.
2. Traffic Controller - which team is best suited to answer this? or to progress this investigation? don’t sit on tickets that can be better served by someone else
3. Custodian (window fixing)

For Every Ticket:

1. Immediately:
   1. Estimate - impact
   2. Mitigate
2. After the event:
   1. Root Cause
   2. Fix
   3. Document on the ticket: root cause and summary of how you fixed it.
3. For open tickets:
   1. Communicate often. Especially if you have a roadblock or if you need to drop the ticket due to priorities.

## Examples

### Good Escalation Email

This email from Marvin is a great example of how to escalate a launch blocker. tl;dr, crisp, clear call to action, AND he had cut a ticket to the appropriate team for the launch team:

**tl;dr - Pickup stores are not being vended to the GLOW bottom-sheet for iOS in the IN marketplace. This is because the React-Native bottom-sheet has not been dialed up in the IN marketplace. See**[**https://t.corp.amazon.com/P39455607**](https://t.corp.amazon.com/P39455607)**.**Hi,Currently the GLOW React-Native bottom-sheet is not dialed up in the IN marketplace for iOS. This is required to vend the pickup stores, as well as support other GLOW features (such as the pickup tab). This is a blocker for the IN-pickup launch. I have opened a ticket to Dexcovery regarding this issue: <https://t.corp.amazon.com/P39455607>. The React-Native bottom-sheet is dialed up in IN for Android.Ticket to F3LIX: <https://t.corp.amazon.com/P39379355>Lindsey – could you please forward this email to the PM/s responsible for the UFG India launch?Thanks,Marvin

# F3lix - Exit Criteria

|  |  |
| --- | --- |
| **Task Type** | **Exit Criteria -** Move out of the Done column only if all the relevant conditions were met: |
| **All** | \* Update the SIM/ticket \* Shared and documented new knowledge in wiki |
| **Scoping** | \* Scoping was shared with Ray and project managers \* The next step is clear. Any required tickets created in backlog. |
| **Development Plan** | \* Include required weblabs \* Include additional required metrics and logging \* Include test plan for integration and E2E tests \* Include additional required HW/AWS Services (if needed) and CPT++ updated, or a scaling to exiting HW/services.  \* Design review done with team and/or DEX \* If QA are engaged, QA review at this point \* All required stories were created in our Kanban Board with Acceptance Criteria |
| **Coding/Bug Fix** | \* Developer tested: Locally, integration tests, Coral, E2E - in Beta and Prod \* Updated the['Current Experience' wiki](https://w.amazon.com/bin/view/F3/F3LIX/GLOW/CurrentExperience/) \* QA notified to test this \* Demo (if relevant) \* Added a ticket for the weblab dialup/down (if relevant) |
| **Weblab** | \* Follow weblabs best practices: <https://w.amazon.com/bin/view/Weblab/Manual/WeblabCodeReviewGuidelines/> \* All related weblabs and MCMs were reviewed and updated \* Added/updated in our weblabs wiki page: <https://w.amazon.com/bin/view/F3/F3LIX/M5/WeblabDialups>  **Dial up:** \* Update our weblabs wiki page (see link above) with the new status and removal date after one month. \* Dependent tasks have been pushed, \* Approvers: \*\* Level 1: the primary that will be called if anything goes wrong (according to dial-up schedule) \*\* Level 2: QA engineer and dex-nyc-lax-primary \*\* Level 3: rasm \* Review comments have been addressed |

## General Notes:

* A task/story can exit if its no longer needed/relevant. In this case, its ticket must be updated to include the change and the reason for it.

# Team Process

# Team Process Proposal

## Tenets (unless you know better ones)

* **Deliver Results:**
  + Our main goal is to deliver the right features in the right quality and on time to our customers.
* **Visibility:** supply visibility for developers, managers and stake holders into:
  + Priorities
  + Ownership: who owns each user-story
  + What each team member is/will be working on in the near future
  + Schedule: when each user-story is expected to be finished
  + Roadblocks and unexpected delays
* **Fairness**:
  + All developers will get to develop real, interesting user-stories
  + All developers share the OE load
* **Light-weight process**:
  + Our process will be as light-weight as possible
  + We will try to use our existing team meetings to support the new process (without adding new ones)
* **Self-correcting**:
  + We will review and adjust our process as part of retro to make sure that it keeps adding value to the team

## Background and Proposal

F3lix has previously used both Kanban and Scrum (in cooperation with POEM). Both systems have advantages and drawbacks. this table try to summarise the main advantages and disadvantages of the two systems, as practiced by us, compared to the tenets above:

|  |  |  |
| --- | --- | --- |
|  | **Kanban** | **Scrum** |
| **Main Advantages** | \* Light-wieght for developers \* Existing, established process | \* Deliverables-based sprints \* Visibility for devleopers and stake hodlers \* Planning allow better division of work and OE \* Less randomisations |
| **Main Disadvantages** | \* Late deliverables (due to all the below) \* Lack of visibility (features schedule, roadblocks, priorities etc) \* Overhead for SDM and PMs (due to lack of visibility) \* Developers randomisation \* Division of work and OE between developers | \* Can become heavy if done 'by the book' (e.g. Estimation meetings etc) \* Can cause burnout if developers are pressured to finish all tasks in a sprint despite roadblocks or inaccurate estimations \* Will need experimentation and tweaking since it's a new process |

I suggest trying the following Scrum-based process for the next 6 months (until October 2021). During this trial, we’ll keep tweaking and improving the process through our retro meetings. At the end of the 6 months, we’ll re-evaluate our process compared with our tenets and discuss if we want to change/replace it.

## Backlog

* All user-stories will live in the backlog, prioritised (by SDM)
* Every user-story will have its due date (if known) in its title (e.g. *[Due: 30.4.21] Some User Story*)
* Every high-priority user-story (in queue for preparing for sprint) will have an owner that will be responsible for it and will act as the story’s POC.
* The owner of each story will scope it, including creating all the required tasks and estimating them. For big user-stories with a lot of unknowns, we might need to spike them in advance (in a different sprint) or to run a POC on them.
* After a story was scoped and briefly discussed with the team, it is ready for sprint.
* Backlog review - the team will review the backlog on a regular basis (on weeklies/planning). We will aim to have a couple of sprints-worth of estimated stories ready to go.

## Estimations

* We will use task points to estimate stories and tasks, similar to the way POEM used them.
* 1 task point == 1 hour of uninterrupted ideal dev work
* 5 task points == 1 day of work
* Stories that are too big to estimate will be spiked in a previous sprint and broken down into smaller stories to be estimated separately.

## Sprints

* We will have 2-weeks sprints
* We will have a P1 and a P2 sprint boards for additional work we can pull into a sprint if we have more capacity and for planning our future sprints
* Every developer will be assigned at least one significant task in each sprint.
* We will have an OE points allocation per sprint and the OE tasks will be divided fairly between the different team members.
* Developers (except for on call) work only on tickets that are on the board. The only exception is that if an urgent task comes in, the developer can start working on it but should bring it up in the next standup and add it to the board.
* Unexpected, urgent tasks will be discussed in standup. If required (for big/many tasks), we will decide which story need to be removed from the sprint.
* We use sprints to improve the predictability of our deliverables and to improve our estimations over time. We don’t pressure developers to finish everything we wanted to in the sprint planning but we commit to working only from the board and to making our estimations more accurate over time.

## Pre Planning

* All team members fill in their capacity for the next sprint ([Team Capacity](https://quip-amazon.com/NTJFA9gkhqzX))
* All team members will punt the tickets they didn’t finish into the next sprint
* The Scrum master will meet with with SDM and/or PMs and define which user-stories need to be done for the next sprint

## Sprint Planning

* Will take into account the capacity of all developers (e.g. vacations, on calls etc)
* On calls capacity will not be counted.
* Secondary capacity will be counted as normal work but will allocate task points for OE tasks and helping the primary.
* We will start with 80% capacity for all developers and adjust it as needed to represent our actual velocity.  
  \*80% is what POEM used successfully as a buffer for inaccurate estimations and small unexpected items.
* Will include defining our sprint demo (see [Demos](https://quip-amazon.com/9OSXAXp61xYX#AbW9CAdxb5c))
* We’ll aim to assign every developer with feature-work and OE task
  + feature work is determined by the business and stake holders
  + OE is driven by the team.

## Standup

* We will use the standup timer (starting with 1 minutes each)
* developers can move their own cards or reduce the remaining task points on them
* Adding task points or new tasks on the board need to be discussed in standup since it might require removing lower-priority tasks or re-assigning tasks to other people.
* We will continue using our post-scrum quip for any long discussions

## Sprint Retro

* Include a normal team retro
* Track and review sprint Burndown Chart and discuss any issues with roadblocks etc. (POEM’s rule of thumb: 80% is expected; 60% there’s a problem)
* How do we track compared to our process tenets?
* What do we want to improve for the next sprint?

## Demos

* Every sprint will have a demo as part of its definition of done. This is a low-weight demo for developers and stake holders only to help us define a sprint goal and demonstrate progress.
* Features that can’t be demoed can still be discussed and explained by the developer

## Open Questions

* OE allocation per sprint? for all devs? for Secondary?
* Meeting-free Mondays?
* Weekly sync and planning - Tuesday or Fridays?
* Estimations for non-trivial stories - in pairs/groups? Start a table with common estimations (e.g. weblab dialup - 3 pts)
* Do we want to do snack names like with POEM?

# 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/

# Mentorship Tips and Tricks

## As Mentor

### To prepare before the meeting

* Have the level guide handy
* Chances are, folk want to know about how to level so know your stuff
* create a quip doc where your mentee can drop questions and you can put links etc

### Questions to ask

what are they most concerned about? what do they want to get out of the mentorship? pointing out the different flavours  
What feedback have they been given by their managers/peers?  
What LP do you want to work on/you don’t understand/can’t relate?  
Have they done a self analysis? what are the gaps in their level?  
How serious and clear are they about their career goals and how serious do those around them perceive them to be?  
What is your biggest technical cock-up so far? What did you learn from it? (Good for judging their response to failure)  
What do you most enjoy doing?

### Advice to give

Wear Sunscreen: <https://www.youtube.com/watch?v=sTJ7AzBIJoI>  
In terms of confidence “Fake it till you make it!”  
Also, most other people also feel like imposters. The only people who don’t are either suffering from Dunning Kruger or are operating in roles grossly below their level.  
tl;dr — use them  
It is more important to succintly answer a persons question than to comprehensively answer their question UNLESS they are asking for a comprehensive answer.  
Do not force a tool to work against its nature.  
Learn the basic unix commandline toolkit.  
Learn how to read log files, learn how to write good log file entries.  
Advice on how to get the most out of 1x1s and feedback  
Read widely, not just computer science or programming.  
All Software is about people. Even if you are writing something to be a beautiful piece of code and nothing more: in that case you are the Customer. By understanding people first you will focus on learning the right things and making your fellow human’s lives better.  
Journally is super effective.  
Do not allow yourself to be distracted by poor sportsmanship of others: what matters is what you do, and that you insist on Highest Standards. Not that they other guy is terrible.  
Sleep. Eat well. Exercise.  
Make room for others to speak, listen to what other people are saying.  
Look after other people, especially when they’re struggling

### Useful Tools/Links to Share

Podcasts:

* coding blocks <https://www.codingblocks.net/category/podcast/>
* software engineerings <https://www.se-radio.net/>
* MS Dev Show - also a very good show on different subject (no just MS-specific): <https://msdevshow.com/>
* Just discovered this one so I’m not sure if it’s any good: <https://www.stitcher.com/podcast/amazon-web-services/aws-podcast>
* this one is just general well being: <https://jimkwik.com/category/podcast/>
* The Pragmatic Programmers
* SDE Good Reads / SDE Tip of Day || Manager Tip of the Week

[https://medium.com/@mrabkin/awkward-1-1s-the-art-of-getting-honest-feedback-2843078b2880](https://medium.com/@mrabkin/the-art-of-the-awkward-1-1-f4e1dcbd1c5c)

## As Mentee

### To prepare before the meeting

What is it that *you* want out of this mentoring engagement. The more specific you can be, the more likely you are to get what you are after. It might help to identify a strength you wish to develop further, or an area for growth you’d like to address, or some common fear you want to work through.  
  
It can help to have a bit of a potted history of yourself ready. “Hi, I am Ray. About a year ago I transitioned to be an SDM. Before that I spent the better part of 20 years as a developer. I’d really like to know how to get Engineers to do anything. It was fun when I was Engineer but now I am the enemy!! Help?!?”  
  
It can be helpful to have a portfolio of work available. Depends on what you want out of the menteeship. But, for example, say you want help to create testable designs. Having some sample design URLs handy is a big help to your Mentor.

### Questions to ask

* what are the most valuable/important things you learned in Amazon?
* What’s the things you wish you knew earlier in your career?
* Did your mentor ever face the same situation that you are concerned about? How did they handle it?
* About other companies. Especially if you haven’t worked as a developer elsewhere.
* Can you tell me about some of your experiences when you were at the same point in your career (that I am now)?
* Tell Me About a Time When <<you displayed the behaviour I want to display myself>>
* Good books to read, book swaps are fun
* What do they like doing and when you can do it together (beer, wings, games, whatever)
* What is your biggest fuck-up? (Every great Engineer has one; adjust language as appropriate)
* How do you find time to up-skill at Amazon?
* I am annoyed about (x), I am going to do (y), does that make sense?
* Not a question but useful thing to talk about: Merhhh I’m wigging out about this thing, it’s making me angry, I’m frustrated
* I have a design/presentation I’m about to do, can we dry-run?

# 