**Technical Phone Screen - Coding**

* Coding Interview Environment

The coding is done 'live' via CoderPad (<http://coderpad.io/>), a collaborative document editor specifically designed for coding phone interviews.

* Language Selection

As there are many different programming languages used at Facebook to solve infrastructure challenges, we don't have a preference for the language (or languages) you use during the interview. We think that if you can show us solid programming skills in one language, you'll be able to transfer them to other languages. At least one of the problems we ask you will require that you use something other than shell scripting (bash/zsh/etc.). We've ended up hiring candidates who have done their phone screens in virtually every modern programming language, including Python, PHP, Perl, Ruby, Java, C++, Go, C#, and Haskell.

* Types of Questions Asked

The questions we ask require skills used every day by Systems Engineers, including text manipulation, handling input/output, automating tasks, interfacing with external systems/processes, etc. The questions can be a real problem, or something contrived to use these skills.

* Interview Tips

Your primary goal in the interview is to obtain a working solution to each problem in a reasonable amount of time.  
  
Make sure you are in a quiet place with a reliable internet connection. Headphones are very handy as holding the phone in your neck for 45 minutes while you try and type is going to be a pain. Close your email and chat apps to avoid interruptions.  
  
We will always paste in the text of any coding question we ask you. Take some time after reading the question to ask questions and plan out your solution, rather than jumping right into its implementation.  
  
Defensive coding is important, but don't focus on details (such as error handling and corner cases) to the detriment of the overall solution. If you're not sure if a given error handling or edge case is important, ask the interviewer.  
  
Don’t get hung up on syntax. If you can’t remember the order or arguments to a function or its name, just say so, leave a placeholder and move on.  
  
While you don’t need to provide a play by play of your thought process throughout the interview, it’s best to let the interviewer know why you are making certain decisions, and do most of your work in the CoderPad window. This will also help with any course corrections that may be needed while you are solving the problem.  
  
Take hints from the interviewer and be open to other solutions as you go. It's totally fine to present a rough solution in the beginning and iterate as you go along.  
  
Use the language you are strongest in. Don't use a language you know less well because it's trendy or you think it will please the interviewer. If the interviewer doesn't know your strongest language, they will figure it out later.  
  
Don’t be afraid to change your mind. If you think you’ve started your solution in the wrong way, or even in the wrong language, it’s OK to admit it and change tack (and the sooner you do it the better).  
  
Resist the temptation to look things up on the internet during your interview. We'd rather have you get a function name incorrect or the order of arguments wrong than have you looking things up during the interview. We want to see you write code and think on your feet, not search the web.

* Preparation

The best way to prepare for coding interviews is to practice under similar circumstances by yourself or with a friend, using sample questions. A coding phone screen is an unnatural event, even if you are used to coding regularly for your job. The problems are different, the environment is different, and you are under time pressure.  
  
Choose at least one language and know the basics solidly. It is better to know one language really well than multiple languages poorly. This includes creating classes and methods, conditional constructs, loops, built-in data structures, input/output, and interfacing with external processes/systems. Be prepared to use this language to solve any type of problem you may get.  
  
Consider practicing by trying coding exercises here: <https://www.interviewbit.com/>   
  
Facebook Engineering   
<https://code.facebook.com/>  
<https://code.facebook.com/posts/>   
<https://www.facebook.com/Engineering>  
Crush Your Coding Interview  
<https://www.facebook.com/Engineering/posts/10152735857917200>

Approach:

**\**How you should approach the interview*\***  
  
\* Ensure that you spend time planning your approach but remember you can always go brute force and then optimize from there.   
\* Articulate/code up a working solution (even if it feels inefficient) and then iterate rather than immediately trying to jump to the clever solution, if you cannot explain your solution clearly in 5 minutes, it's probably too complex for the problem at hand.   
\* Make sure you are asking clarifying questions as you go along (there will not be tricks but you will need to ensure you have all the info you need).   
\* Practice talking through your coding out loud while you are doing it - the interviewers need to see where you are headed with your solutions and if you talk they can provide hints.  
\* Spend a little time thinking about how you would walk someone else through your past projects, what you did on them (as opposed to what your team did), and what you learned from them.   
\* Come prepared with questions for your interviewer - this is just as much a chance for you to ensure that this is the place you want to be.  
\* Make sure you use good test cases that will find edge/corner cases.  
\* You will need to debug your code.  
  
**\**What content you should know*\***  
  
Spend time brushing up on your CS fundamentals – algorithms, data structures, etc.   
  
\* You will need to demonstrate practical mastery of the following. -  
\* Arrays and lists, binary trees, hash tables, stacks and queues, basic graph representations; an understanding of data structure fundamentals like pointers (as if you weren't using an object oriented language).  
\* Make sure you have both time and space complexity down for the data structures you end up using in your solutions – become fluent in Big O Notation; knowing O(n), O(1), etc is important.  
\* Also look over search through data structures - understand how to implement iterators, understand binary search, and how hashes work, have your sorts down - merge, quick, bucket sorts, know how to traverse a graph (either BFS or DFS), and understand how to do recursions. You will need to write these out and explain them to another person on a white board so practice this!  
  
[Here is a graphic](http://www.crackingthecodinginterview.com/uploads/6/5/2/8/6528028/handout_-_cracking_the_coding_skills.png) to summarize the above info.  
  
[A Quora answer](https://www.quora.com/How-do-I-prepare-for-a-software-engineering-job-interview) on how to prep for a technical interview.