

# Technical Interviews

*practicing the essential skills to ace technical interviews*

**CSEC**

# Itinerary

---

Time	Event
4:00	The Hub Opens
<b>4:15</b>	<b>Event Starts</b>
4:50	Break and Refreshments
5:10	Interviews – Mentor 1
6:00	Interviews – Mentor 2
6:50	Wrap up and Final Words

# The Technical Interview

---

Most companies in other disciplines use behavioral interviews to screen candidates

Tech companies use 'Technical Interviews'

- Tests understanding of fundamental CS knowledge
- Tests ability to reason through difficult problems
- Tests the ability to adapt to changing requirements
- Tests the understanding of edge cases and testing



# The Technical Interview

---

Technical screens come in many different forms:

- Take-home coding evaluations/challenges
- Brain Teasers
- Past projects/experience/jobs
- Hidden puzzles (e.g. Google, Uber)
- **Technical Questions**

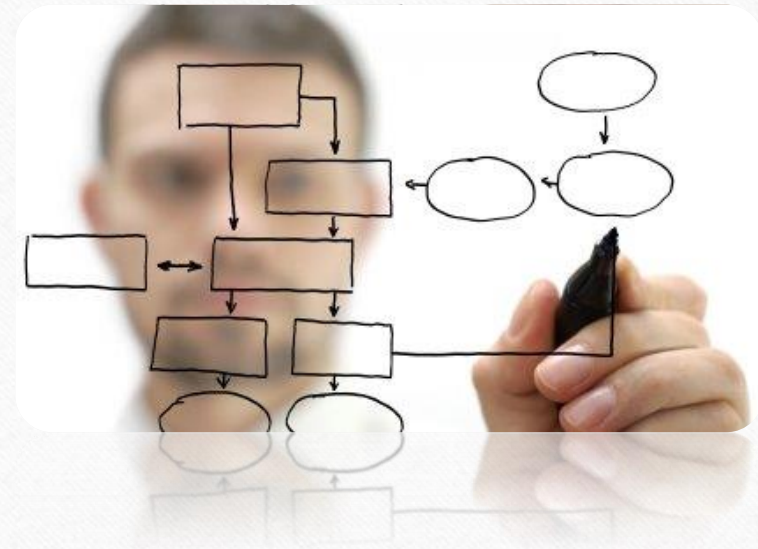


# Technical Questions

---

**Technical Questions** test candidates on a variety of different knowledge. They can usually be generalized into:

- **Basic Knowledge**
- **Algorithmic Questions**
- **Data Structure Questions**
- **Design Questions**
- **Tech-specific Questions**



# Basic Knowledge

---

## Basic Knowledge

Questions are supposed to be competency tests for most developers. These questions take a look at a candidate's:

- **Correct interpretation of problem to solve**
- **Confidence in programming language**
- **Ability to work through examples**
- **Not overcomplicate**
- **Address edge cases**

# Basic Knowledge

---

## **FizzBuzz**

*Infamous 'candidate competency screening test'*

“Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”

# FizzBuzzEnterpriseEdition

build passing codecov 89%

Enterprise software marks a special high-grade class of software that makes careful use of relevant software architecture design principles to build particularly customizable and extensible solutions to real problems. This project is an example of how the popular FizzBuzz game might be built were it subject to the high quality standards of enterprise software.

161 commits

1 branch

0 releases

30 contributors

Branch: master



New pull request














Create new file

Upload files

Find file

Clone or download

 **emiln** committed on **GitHub** Merge pull request #281 from tkellogg/master  Latest commit `cdfac75` on Feb 4

 <code>gradle/wrapper</code>	Include Gradle config <a href="#">closes #230</a>	2 years ago
 <code>resources/assets/configuration/sprin...</code>	Installed spring and restructured for dependency injection.	3 years ago
 <code>src</code>	FizzStringReturner may not have copied all characters	9 months ago
 <code>.gitattributes</code>	Started the project. Time to learn!	5 years ago
 <code>.gitignore</code>	Add unsupported platform files to the .gitignore	2 years ago
 <code>.travis.yml</code>	Reverting the revert commit, since it clearly did not revert as inten...	2 years ago
 <code>CONTRIBUTING.md</code>	Reverting the revert commit, since it clearly did not revert as inten...	2 years ago
 <code>README.md</code>	Reverting the revert commit, since it clearly did not revert as inten...	2 years ago
 <code>build.gradle</code>	Include Gradle config <a href="#">closes #230</a>	2 years ago
 <code>gradlew</code>	Include Gradle config <a href="#">closes #230</a>	2 years ago
 <code>gradlew.bat</code>	Include Gradle config <a href="#">closes #230</a>	2 years ago
 <code>pom.xml</code>	Merge pull request #232 from kristianperkins/patch-1	a year ago
 <code>settings.gradle</code>	Include Gradle config <a href="#">closes #230</a>	2 years ago



# FizzBuzz – In Python

---

“Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”

```
for x in range(0, 100):  
    if(x % 3):  
        print("Fizz")  
    if(x % 5):  
        print("Buzz")  
    if(x % 3 && x % 5):  
        print("FizzBuzz")
```

# FizzBuzz – In Python

---

“Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”

```
for x in range(0, 100): (1, 101):
    if(x % 3):           x % 3 == 0, add elif
        print("Fizz")
    if(x % 5):           x % 5 == 0, add elif
        print("Buzz")
    if(x % 3 && x % 5):   Put above both, use 'and'
        print("FizzBuzz")
    else:
        print(x)
```

# FizzBuzz – In Python

---

“Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”

```
for x in range(1, 101):
    if(x % 3 == 0 and x % 5 == 0):
        print("FizzBuzz")
    elif(x % 3 == 0):
        print("Fizz")
    elif(x % 5 == 0):
        print("Buzz")
    else:
        print(x)
```

# Algorithmic Questions

---

## Algorithmic Questions

Questions that test a candidate's ability to identify and apply the correct algorithm(s) to a problem.  
Try to focus on:

- **Defining and working through examples**
- **Thinking about edge cases**
- **Identifying a brute-force solution**
- **Identifying and implementing a better algorithm**
- **Analyzing the runtime of the algorithm**

# Algorithmic Questions

---

## Reversing Strings

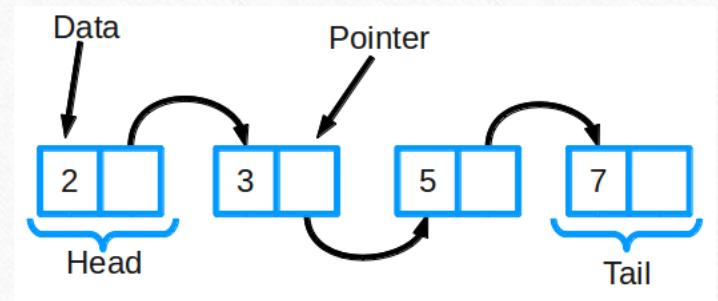
“Given a string, `s`, reverse `s` in place without using any built-in functions that would trivialize the solution”

# Data Structures Questions

## Data Structures Questions

Questions that test a candidate's knowledge of various data structures and their usage. Candidates should be expected to be familiar with:

- **When to use different data structures**
- **Unique properties of each data structure**
- **Implementation of basic data structures and operations**
- **Runtime of basic operations on data structures**

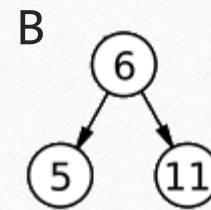
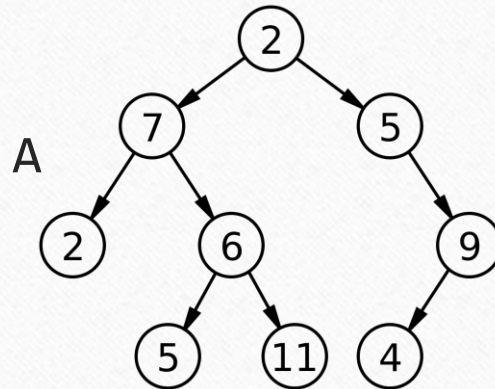


# Data Structures Questions

---

## Binary Subtree Detection

“Given a binary tree, A, determine if a smaller binary tree B is a subtree of A”



**Expected Output:**

True

# Design Questions

---

## Design Questions

Questions that test a candidate's ability to plan and design scalable systems that follow SOLID design principles. Candidates should focus on:

- **Identifying the core issue to address**
- **Modelling Relationships and making Design Decisions**
- **Object Oriented Design**
  - SOLID Design principles
  - Object-oriented programming principles (Polymorphism, Encapsulation)
  - Various design patterns (Factory, Observer, Singleton, etc.)
- **Scalability and extensibility of solution**



# Design Questions

---

## Connect Four

“Expect that someone will handle all of the front end and user interface. Please implement the game of Connect Four.”



# Tech-Specific Questions

---

## Tech-Specific Questions

These questions are usually looked down upon because they don't exactly test how strong a candidate is, rather just if they ever used something. Nonetheless, they do come up. Some examples could be:

- What are *virtual functions* in C++?
- What does *volatile* do in C?
- What are *lambdas* in Python?
- What is the difference between *final* and *finally* in Java?
- What is the difference between a *hardlink* and a *symlink* in UNIX?

# Practice and Resources

---

## Technical Interview Books:

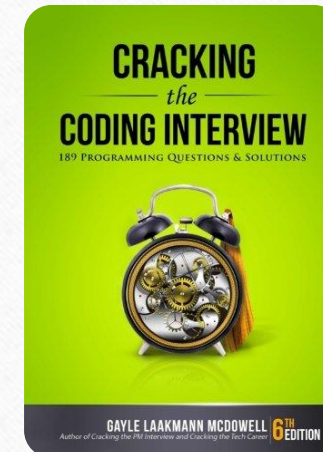
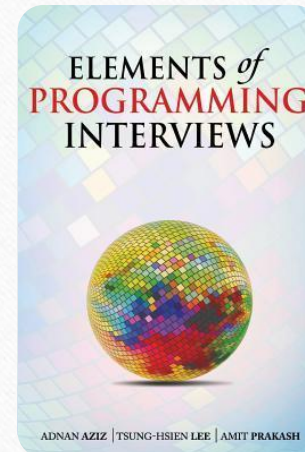
- Cracking the Coding Interview
- Programming Interviews Exposed
- Elements of Programming Interviews

## Question Practice:

- LeetCode
- HackerRank

## Mock Interviews

- Pramp



# Itinerary

---

Time	Event
4:00	The Hub Opens
4:15	Event Starts
<b>4:50</b>	<b>Break and Refreshments</b>
5:10	Interviews – Mentor 1
6:00	Interviews – Mentor 2
6:50	Wrap up and Final Words

# Itinerary

---

Time	Event
4:00	The Hub Opens
4:15	Event Starts
4:50	Break and Refreshments
<b>5:10</b>	<b>Interviews – Mentor 1</b>
6:00	Interviews – Mentor 2
6:50	Wrap up and Final Words

# Itinerary

---

Time	Event
4:00	The Hub Opens
4:15	Event Starts
4:50	Break and Refreshments
5:10	Interviews – Mentor 1
<b>6:00</b>	<b>Interviews – Mentor 2</b>
6:50	Wrap up and Final Words

# Itinerary

---

Time	Event
4:00	The Hub Opens
4:15	Event Starts
4:50	Break and Refreshments
5:10	Interviews – Mentor 1
6:00	Interviews – Mentor 2
<b>6:50</b>	<b>Wrap up and Final Words</b>