# Introduction

Introduce yourself and state that you work for Amazon, state why you are calling, ask for the person by name.

Confirm the time is correct, the number is acceptable (they may prefer a landline over a cell phone), and that they are ready to do the phone interview.

The format of the following script is:
**Question the interviewer should ask in bold.**
General criteria for evaluating their response follow in normal text. Most have vague descriptions of characteristics of various qualities of answers.
Excellent answers will include such and such.
Good answers will include a lesser list of attributes.
Poor answers will conform to some particular template.

# Culture Fit

**Why are you leaving your current position?**
A good answer will be generally positive, seeking better challenges or goals. Contract expirations and familial obligations are also common.
A poor answer would be ranting against the current employer or generally being negative. Honesty is not necessarily negativity. The key is how they view their departure, whether they seek self-improvement whatever the situation.

**Why do you want to work for Amazon?**
An excellent answer will be enthusiastic about Amazon and its business and problem space, demonstrating knowledge of our site and familiarity with its use.
A good answer will be enthusiastic about Amazon and show a desire to work in this environment and on our sorts of problems.
A mediocre answer will demonstrate a lack of knowledge of our site and operations

**What was your favorite project?**
An excellent answer will discuss an in-depth project involving a complicated problem, have a clear description of the problem that explains why it was difficult, describe the solution chosen, and reference how the candidate was involved in the entire process and implementation.
A good answer may involve a small project that was well-executed, or a large project working on a simple problem, or a complex problem for which a sub-optimal solution was chosen.
An acceptable answer will describe a project in detail from problem to implementation.
A poor answer will not give a clear explanation of what the purpose of the project was, or will neglect to mention a solution at all, or will say the candidate was not actually involved in the project being described.

**Tell me about your least favorite project.**

**What do you look for in a job?**

**How do you feel about working for a book store?**

**What is your understanding of how Amazon does business?**

## Project History

**What classes have you taken?**

**What projects have you worked on?**

**Describe a difficult problem you had to solve.**

**Describe some technologies you've worked with.  (dig into the technology)**

## Data Structures

### *First Phone Screen*

All these questions should be asked.

**What is a linked list?**
A competent person will know what a linked list is. This is pass/fail.

**Compare a linked-list to a vector.**
An excellent answer will discuss the implementation differences, different pattern of memory usage and access, and mention some use cases where one is more appropriate than the other.
A good answer will discuss the implementation differences and elaborate on some examples.
An acceptable answer will merely mention the access time differences.
A poor answer will make a technical mistake or not know one or both data structures.

**What is a hash table?  What is it used for?**
An excellent answer will describe an O(1) read and write data structure with scaling memory usage and give several examples of where it is useful (e.g. caching, lookup tables with irregular keys).
A good answer will describe an O(1) read and write data structure and give an example of where it is useful.
An acceptable answer will mention constant-time access.

A poor answer will give the access time as something other than constant-time, or confuse it with a tree or other structure.

**How can a Map data structure be implemented?**
By following the question regarding hash tables, this should be a gimme.
An excellent answer will cite at least the hash table-backed and the tree-backed implementations and note the different performance characteristics of each, possibly mentioning that the tree-backed implementation can provide some features not available in a hash-backed implementation (e.g. sorting).
A good answer will cite both hash table and tree implementations.
An acceptable answer will cite either the hash table or tree implementation.
Being unable to answer this question is bad.

**What is a binary tree?**
This question follows the Map question to allow people who forgot non-hash table solutions to realize and revise their omission. It is good for candidates who only mentioned the hash table implementation of a map to point out that a tree is also an alternative implementation after this question is asked.
An excellent answer will describe a binary tree as distinct from a ternary or general tree, and will note that a binary search tree or balanced tree are distinct sub-categories of the generic binary tree.
A good answer will describe a generic tree where every node has no more than two children and will cite the O(log N) access time where N is the depth/height of the tree.
An acceptable answer will describe a tree where every node has no more than two children.
A poor answer will respond by describing a binary search tree or a non-binary tree, and will not respond to gentle hinting or prodding back to the specific question that was asked.

## *Second Phone Screen*

At least four or five of the following or similar questions should be asked.

**What is operational complexity?**

**What is the operational complexity of inserting an element into a linked list?**
Asking a clarifying question about singly-linked lists versus doubly-linked lists is unnecessary in this case, but clarifying requirements is always a good thing.
An excellent answer will correctly cite O(N) and note the edge cases such as empty lists and the beginning/ending of the list.
An acceptable answer will correctly cite O(N).
A poor answer will be incorrect.

**What is the operational complexity of inserting an element into a hash table?**

**How is a hash table implemented?**

An excellent answer will discuss selecting a hashing algorithm, dividing the address space into buckets and tuning the number of buckets to the use-case, hash collisions and bucket overflow/conflict resolution (possibly mentioning several strategies for dealing with collisions), dynamic resizing of the hash table, and might even discuss performance optimizations and interactions with hardware caching.
A good answer will note the use of a hashing algorithm on the keys and an array or vector of buckets, and will identify hash collisions as a problem and may cite at least one method of dealing with them. The candidate will be able to answer a follow-up about dynamic resizing.
An acceptable answer will mention an array of buckets and using a hashing function on the key to index into the array.
A poor answer will be technically incorrect or impossible, or will be for a different data structure than a hash table.

**What is the operational complexity of (various operations) on a binary tree?**
If the candidate already answered this as part of the first phone screen, this question may be skipped. It may also be asked to ensure the candidate is consistent in responding correctly.

# Coding / Problem Solving

## *First Phone Screen*

**Implement an algorithm to covert a string into a double.**

**Reverse a linked list.**

**Determine whether a linked list is cyclical.**

**Implement an algorithm to determine the angle between the hands of a clock at a specific time.**

**Given a deck of 52 cards numbered from 1 to 52, determine which card is missing.**

## *Second Phone Screen*

**Given two linked lists, determine whether they intersect at any point.**

**In a doubly-linked binary tree, write an algorithm to determine whether two nodes are part of the same tree.**

**In a doubly-linked binary tree, find the nearest common ancestor.**

**Find the shortest route between two nodes of a graph.**

**Implement a method to detect whether two time intervals intersect.**

## Object Oriented Design

### *First Phone Screen*

Ask all the following questions.

**What is inheritance?**

**What is polymorphism?**

**What is encapsulation?**

**What is multiple inheritance?**

**How is inheritance implemented in your favorite language?**

**What is the difference between OO programming and structured programming?**

**What is a design pattern?**

**Describe your favorite design pattern.**

**What is a Singleton?**

**Design a class to represent an interval of time. (mutable vs. immutable, what operations should it support).**

### *Second Phone Screen*

Ask four of the following questions, including at least one regarding designing an object model. Dig deeper on their design and challenge their assumptions with edge cases.

**Describe the object model of a supermarket.**

**Describe the object model of a chess board.**

**Describe the object model of a parking lot.**

**Describe the object model of a deck of cards.**

**What is a Visitor pattern?**

**What is the Factory pattern?**

**How does a Factory differ from a Singleton?**

**What is the Decorator pattern?**

# UNIX

## *First Phone screen*

How do you know what directory you’re in?

How do you change directories?

How do you create a new directory?

How do you know what processes are running on a machine?

How do you know how much drive space is left?

What does “top” do?

How would you figure out what unique IP addresses are recorded in a web server log file?

What is a pipe?

What’s the difference between a symbolic link and a hard link?

## *Second Phone Screen*

What’s the different between “ssh” and “telnet”?

What’s your favorite UNIX editor?

What’s your favorite shell?

How do you change the default settings of your shell?

How would you convert the stdout of one command into the command-line arguments of another command?

How do you convert command-line arguments into stdout?

How do you run a command on multiple files in a directory tree?

What scripting languages do you know?