1. What is a Data Structure?

- A data structure is a storage format that defines the way data is stored, organized, and manipulated.
- Some popular data structures are Arrays, Trees, and Graphs

2. What is an Array?

- An array is commonly referred to as a collection of items stored at contiguous memory locations.
- Items stored are of the same type.
- It organizes data so that a related set of values can be easily sorted or searched

3. What is a Linked List?

- Like an array, a linked list refers to a linear data structure in which the elements are not necessarily stored in a contiguous manner.
- It is basically a sequence of nodes, each node points towards the next node forming a chain-like structure

4. What is LIFO?

- LIFO is an abbreviation for Last In First Out
- It is a way of accessing, storing and retrieving data.
- It extracts the data that was stored last first.

5. What is a Stack?

- A stack refers to a linear data structure performing operations in a LIFO (Last In First Out) order.
- In a stack, elements can only be accessed, starting from the topmost to the bottom element.

6. What is FIFO?

- FIFO stands for First In First Out.
- It is a way of accessing, storing and retrieving data.
- The data that was stored first is extracted first

7. What is a Queue?

- A queue refers to a linear data structure that performs operations in a FIFO order.
- In a queue, the least recently added elements are removed first as opposed to a stack.

8. What are Binary Trees?

- A binary tree is an extension of the linked list structure where each node has at most two children.
- A binary tree has two nodes at all times, a left node and a right node

9. What is Recursion?

- Recursion refers to a function calling itself based on a terminating condition.
- It uses LIFO and therefore makes use of the stack data structure.

The next couple of coding interview questions will explore your knowledge of OOPs.

## 10. What is the OOPs concept?

OOPs stands for Object-Oriented Programming System, a paradigm that provides concepts such as objects, classes, and inheritance.

## 11. What are the concepts introduced in OOPs?

Following are the concepts introduced in OOPs:

- Object - A real-world entity having a particular state and behavior. We can define it as an instance of a class.
- Class - A logical entity that defines the blueprint from which an object can be created or instantiated.
- Inheritance - A concept that refers to an object gaining all the properties and behaviors of a parent object. It provides code reusability.
- Polymorphism - A concept that allows a task to be performed in different ways. In Java, we use method overloading and method overriding to achieve polymorphism.
- Abstraction - A concept that hides the internal details of an application and only shows the functionality. In Java, we use abstract class and interface to achieve abstraction.

- Encapsulation - A concept that refers to the wrapping of code and data together into a single unit.

This is one of the very common coding interview questions, that often allows the interviewer to branch out into related topics based on the candidate’s answers

## 12. Explain what a Binary Search Tree is.

- A binary search tree is used to store data in a manner that it can be retrieved very efficiently.
- The left sub-tree contains nodes whose keys are less than the node’s key value.
- The right sub-tree contains nodes whose keys are greater than or equal to the node’s key value

## 13. Explain Doubly Linked Lists?

- Doubly linked lists are categorized as a special type of linked list in which traversal across the data elements can be done in both directions.
- This is made possible by the presence of two links in every node, one that links to the node next to it and another that connects to the node before it.

## 14. What is a Graph?

- A graph is a particular type of data structure that contains a set of ordered pairs.
- The ordered pairs in a graph are also known as edges or arcs and are most commonly used to connect nodes where the data can be stored and retrieved.

15. Differentiate between linear and non-linear data structure?

| ***Linear data structure*** | ***Non-linear data structure*** |
|---|---|
| It is a structure in which data elements are adjacent to each other | It is a structure in which each data element can connect to over two adjacent data elements |
| Examples of linear data structure include linked lists, arrays, queues, and stacks | Examples of nonlinear data structure include graphs and trees |

16. What is a Deque?

- A deque is a double-ended queue.
- This is a structure in which elements can be inserted or removed from either end.

17. What's the difference between Stack and Array?

| Stack | Array |
|---|---|
| Stack follows a Last In First Out (LIFO) pattern. What this means is that data access necessarily follows a particular sequence where the last | On the other hand, Arrays do not follow a specific order, but instead can be accessed |

| | |
|---|---|
| data to be stored is the first one that will be extracted. | or called by referring to the indexed element within the array. |

18. Which sorting algorithm is the best?

- There are many types of sorting algorithms: bubble sort, quick sort, balloon sort, merge sort, radix sort, and more.
- No algorithm can be considered as the best or fastest because they have designed each for a specific type of data structure where it performs the best

19. How does variable declaration affect memory?

- The amount of memory that is to be reserved or allocated depends on the data type being stored in that variable.
- For example, if a variable is declared to be “integer type”, 32 bits of memory storage will then be reserved for that particular variable.

20. What are dynamic data structures?

Dynamic data structures have the feature where they expand and contract as a program runs. It provides a very flexible method of data manipulation because adjusts based on the size of the data to be manipulated.

These 20 coding interview questions that test the conceptual understanding of the candidates give the interview a clear idea on how strong the candidate’s fundamentals are

## Programming Interview Questions

The next set of coding interview questions focus tests the programming expertise of the candidates and dives deep into various related aspects.

The code screenshots given along with the below coding interview questions helps you provide the answer to the question, with clarity.

### 21. How do you reverse a string in Java?

- Declare a string.
- Take out the length of that string.
- Loop through the characters of the string.
- Add the characters in reverse order in the new string.

```
String str = "hello";

String reverse = "";

int length = str.length();

for (int i = 0; i < length; i++) {

    reverse = str.charAt(i) + reverse;

}

System.out.println(reverse);
```

22. How do you determine if a string is a palindrome?

- A string is a palindrome when it stays the same on reversing the order of characters in that string.
- It can be achieved by reversing the original string first and then checking if the reversed string is equal to the original string.

```
if (str.equals(reverse)) {

    System.out.println("Palindrome");

} else {

    System.out.println("Not Palindrome");

}
```

23. Find the number of occurrences of a character in a String?

To find the number of occurrences, loop through the string and search for that character at every iteration; whenever it is found, it will update the count.

```
int count = 0;

char search = 'a';

for (int i = 0; i < length; i++) {

    if (str.charAt(i) == search) {

        count++;
```

```
   }

}

System.out.println(count);
```

24. How to find out if the given two strings are anagrams or not?

Two strings are anagrams if they contain a similar group of characters in a varied sequence.

- Declare a boolean variable that tells at the end of the two strings are anagrams or not.
- First, check if the length of both strings is the same, if not, they cannot be anagrams.
- Convert both the strings to character arrays and then sort them.
- Check if the sorted arrays are equal. If they are equal, print anagrams, otherwise not anagrams.

```
boolean anagrmstat = false;

if (str.length() != reverse.length()) {

    System.out.println(str + " and " + reverse + " not anagrams string");

} else {

    char[] anagram1 = str.toCharArray();

    char[] anagram2 = reverse.toCharArray();
    Arrays.sort(anagram1);

    Arrays.sort(anagram2);

    anagrmstat = Arrays.equals(anagram1, anagram2);

}

if (anagrmstat == true) {

    System.out.println(" anagrams string");

} else {

    System.out.println(" not anagrams string");

}
```

25. How do you calculate the number of vowels and consonants in a String?

- Loop through the string.
- Increase the vowel variable by one whenever the character is found to be a vowel, using the if condition. Otherwise, increment the consonant variable.
- Print the values of both the vowel and the consonant count.

```
int vowels = 0;

int consonants = 0;

for (int k = 0; k < str.length(); k++) {

    char c = str.charAt(k);
    if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u')

        vowels++;

    else

        consonants++;

}

System.out.println("Vowel count is " + vowels);

System.out.println("Consonant count is: " + consonants);
```

26. How do you get the matching elements in an integer array?

- Declare an array.
- Nest a couple of loops to compare the numbers with other numbers in the array.
- Print the matching elements if found.

```
int[] a = { 1, 2, 3, 4, 5, 1, 2, 6, 7 };

for (int m = 0; m < a.length; m++) {

    for (int n = m + 1; n < a.length; n++) {

        if (a[m] == a[n])

            System.out.print(a[m]);

    }

}
```

27. How would you implement the bubble sort algorithm?

- Declare an array.
- Nest a couple of loops to compare the numbers in the array.
- The array will be sorted in ascending order by replacing the elements if found in any other order.

```
int[] a = { 1, 2, 7, 6, 4, 9, 12 };

for (int k = 0; k < a.length; k++) {

    for (int l = 0; l < a.length - l - 1; l++) {

        if (a[l] > a[l + 1]) {

            int t = a[l];

            a[l] = a[l + 1];

            a[l + 1] = t;

        }

    }

}
```

28. How would you implement the insertion sort algorithm?

- We assume the first element in the array to be sorted. The second element is stored separately in the key. This sorts the first two elements. You can then take the third element and do a comparison with the ones on the left of it. This process will go on until a point where we sort the array.

```
int[] a = { 1, 2, 7, 6, 4, 9, 12 };

for (int m = 1; m < a.length; m++) {

    int n = m;

    while (n > 0 && a[n - 1] > a[n]) {

        int k = a[n];

        a[n] = a[n - 1];

        a[n - 1] = k;

        n--;

    }

}
```

29. How do you reverse an array?

- Loop till the half-length of the array.
- Replace the numbers corresponding to the indexes from the starting and the end.

```
int[] a = { 1, 2, 7, 6, 4, 9, 12 };

for (int t = 0; t < a.length / 2; t++) {

    int tmp = a[t];

    a[t] = a[a.length - t - 1];
    a[a.length - t- 1] = tmp;

}
```

30. How would you swap two numbers without using a third variable?

- Declare two variables and initialize them with values.
- Make b the sum of both numbers.
- Then subtract the sum (b) from a, so a is now swapped.
- Lastly, subtract a from the sum (b), so b is also swapped.

```
int a = 10;

int b = 20;

b = b + a; // now b is sum of both the numbers

a = b - a; // b - a = (b + a) - a = b (a is swapped)

b = b - a; // (b + a) - b = a (b is swapped)
```

31. Print a Fibonacci series using recursion?

- The Fibonacci numbers are the numbers in the following integer sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

- We can calculate them using the mathematical formula used in the Fibonacci recursive function.

```
public static int fibonacci(int n) {
    if (n <= 1)

        return n;

    return fibonacci(n - 1) + fibonacci(n - 2);

}

public static void main(String args[]) {

    int n = 10;

    System.out.println(fibonacci(n));

}
```

32. How do you find the factorial of an integer?

- A factorial is a function that multiplies a number by every number below it. For example, 5!= 5*4*3*2*1=120.
- Recursive function multiples the numbers until it reaches 1.

```
public static long factorial(long n) {

if (n == 1)

    return 1;

else

    return (n * factorial(n - 1));

}
```

33. How do you reverse a Linked List?

- Declare a linked list.
- Add elements to that linked list.
- Apply the descending iterator method to the linked list.
- This reverses the order of elements in the linked list.

```
LinkedList<Integer> ll = new LinkedList<>();

ll.add(1);

ll.add(2);

ll.add(3);

System.out.println(ll);

LinkedList<Integer> ll1 = new LinkedList<>();

ll.descendingIterator().forEachRemaining(ll1::add);

System.out.println(ll1);
```

34. How would you implement Binary Search?

- Binary search divides the array into half in every iteration step until it finds the element.
- It works on the sorted arrays since it compares the values of adjacent elements and then calculates the mid number.
- If the value of low becomes greater than high at any point, it means the element is not present in the list.

```
int mid = (low + high) / 2;
```

```
while (low <= high) {
    if (arr[mid] < key) {
        low = mid + 1;
    } else if (arr[mid] == key) {
        return mid;
    } else {
        high = mid - 1;
    }
    mid = (low + high) / 2;
}
if (low > high) {
    return -1;
}
return -1;
```

35. How would you find the second largest number in an array?

- Loop through the array.
- If the value of i is greater than the highest, store the value of i in highest, and store the value of highest in the second-highest variable.

```
private static int findSecondHighest(int[] array) {

    int highest = Integer.MIN_VALUE;

    int secondHighest = Integer.MIN_VALUE;

    for (int i : array) {

        if (i > highest) {

            secondHighest = highest;

            highest = i;

        } else if (i > secondHighest) {

            secondHighest = i;

        }

    }

    return secondHighest;

}
```

36. How do you remove all occurrences of a given character from the input string?

- Use the built-in string method “replace” to replace a character with any other character, including symbols and white spaces.

```
String str1 = "Australia";

str1 = str1.replace("a", "");
```

```
System.out.println(str1); // ustrli
```

37. Showcase Inheritance with the help of a program?

- The class Cat inherits the property color from the class Animal by extending the parent class (Animal).
- This way a class Cat can have more parent classes if it wishes to inherit their properties.

```
class Animal {

    String color;

}

class Cat extends Animal {

    void meow() {

        System.out.println("Meow");

    }

}
```

38. Explain overloading and overriding with the help of a program?

Overloading:

When a class has two or more methods with the same name, they are called overloaded methods.

```
class Foo {
    void print(String s) {

        System.out.println(s);

    }

    void print(String s, int count) {

        while (count > 0) {

            System.out.println(s);

            count--;

        }

    }

}
```

Overriding:

When a superclass method is also implemented in the child class, it’s a case of overriding.

```
class Base {

    void printName() {

        System.out.println("Base Class");

    }

}
```

```
class Child extends Base {

    @Override

    void printName() {

        System.out.println("Child Class");

    }

}
```

39. How do you check if the given number is prime?

- Use if statements to check for each condition separately:
  - If the number is 0 or 1, it cannot be prime.
  - If the number is 2, it is prime number.
  - If the number is indivisible by other numbers, it is prime.

```
public static boolean isPrime(int n) {

    if (n == 0 || n == 1) {

        return false;

    }

    if (n == 2) {

        return true;
    }

    for (int i = 2; i <= n / 2; i++) {

        if (n % i == 0) {

            return false;

        }

    }

    return true;

}
```

40. How do you sum all the elements in an array?

- Use for loop to iterate through the array and keep adding the elements in that array.

```
int[] array = { 1, 2, 3, 4, 5 };

int sum = 0;

for (int i : array)

    sum += i;

System.out.println(sum);
```

1. How do you reverse a string?
2. How do you determine if a string is a palindrome?
3. How do you calculate the number of numerical digits in a string?

4. How do you find the count for the occurrence of a particular character in a string?
5. How do you find the non-matching characters in a string?
6. How do you find out if the two given strings are anagrams?
7. How do you calculate the number of vowels and consonants in a string?
8. How do you total all of the matching integer elements in an array?
9. How do you reverse an array?
10. How do you find the maximum element in an array?
11. How do you sort an array of integers in ascending order?
12. How do you print a Fibonacci sequence using recursion?
13. How do you calculate the sum of two integers?
14. How do you find the average of numbers in a list?
15. How do you check if an integer is even or odd?
16. How do you find the middle element of a linked list?
17. How do you remove a loop in a linked list?
18. How do you merge two sorted linked lists?
19. How do you implement binary search to find an element in a sorted array?
20. How do you print a binary tree in vertical order?
21. What is a data structure?
22. What is an array?
23. What is a linked list?
24. What is the difference between an array and a linked list?
25. What is LIFO?
26. What is FIFO?
27. What is a stack?
28. What are binary trees?
29. What are binary search trees?
30. What is object-oriented programming?
31. What is the purpose of a loop in programming?
32. What is a conditional statement?

33. What is debugging?
34. What is recursion?
35. What are the differences between linear and non-linear data structures?
36. What programming languages do you have experience working with?
37. Describe a time you faced a challenge in a project you were working on and how you overcame it.
38. Walk me through a project you're currently or have recently worked on.
39. Give an example of a project you worked on where you had to learn a new programming language or technology. How did you go about learning it?
40. How do you ensure your code is readable by other developers?
41. What are your interests outside of programming?
42. How do you keep your skills sharp and up to date?
43. How do you collaborate on projects with non-technical team members?
44. Tell me about a time when you had to explain a complex technical concept to a non-technical team member.
45. How do you get started on a new coding project?

## Technical Coding Interview Questions [Programming-Based]

We'll start with the common questions that are used to evaluate your foundational knowledge of coding. Depending on the interview, you may be expected to present your answers vocally, write them on a board, or use a provided laptop.

String

The following questions assess your understanding of the "string" data type.

***How Do You Reverse a String?***

There are many methods available to reverse a string in Python, for example:

- Slicing
- Use join
- Loop

The below Python snippet shows an example of the loop method being used to iterate characters in reverse order and then append them.

```
def reverse(myString):
   reversedString = ''
   index = len(myString)
   while index:
       index -= 1
       reversedString += myString[index]
   return reversedString
```

***How Do You Calculate the Number of Vowels and Consonants in a String?***

1. Create a lower-case copy of the string to simplify the checks.
2. Loop through each character in the string.
3. Check against each vowel, as the set is small.
4. If it’s not a vowel, check if it is within the ASCII table values of all known letters (as a character is ultimately a numerical value).

```
void CountVowelsAndConsonants(string input, out int
numVowels, out int numConsonants)
{
   numVowels = 0;
   numConsonants = 0;
   var loweredInput = input.ToLower();

   foreach (var c in loweredInput)
   {
       if (c == 'a' || c == 'e' || c == 'i' || c == 'u')
           numVowels++;
```

```
        else if (c >= 'a' && c <= 'z')
            numConsonants++;
    }
```

***How Do You Print the First Non-Repeated Character in a String?***

1. Create an array of 256, the amount of valid ASCII characters, and initialize each value to -1.
2. Loop through each character in the input string.
3. Using the character as an index in the indices array, if the value is -1, you are seeing it for the first time. In that case, set it to the current loop index.
4. If the value isn’t -1, you have processed this character before, so mark this with -2.
5. Loop through the indices array and find the smallest value (this is the index in the string), this is your first non-repeating character.

```
int[] indices = new int[256];

for (int i = 0; i < 256; i++)
   indices[i] = -1;

for(int i = 0; i < input.Length; i++)
{
   var c = input[i];
   if (indices[c] == -1)
       indices[c] = i;
   else
       indices[c] = -2;
}

int firstNonRepeatingIndex = Int32.MaxValue;
foreach (var index in indices)
   if (index >= 0)
       firstNonRepeatingIndex =
Math.Min(firstNonRepeatingIndex, index);
```

***How Do You Figure Out if the Provided String Is a Palindrome?***

A palindrome is a string of characters that retains the same order whether it is written forward or backward. You can find out if a string is a palindrome by reversing it and then comparing the two versions as shown below:

```
bool IsPalindrome(string input)
{
   var inputCharArray = input.ToCharArray();
   Array.Reverse(inputCharArray);
   var reversedInput = new string(inputCharArray);
   var isPalindrome = string.Equals(reversedInput, input,
StringComparison.CurrentCultureIgnoreCase);

   return isPalindrome;
}
```

***How Do You Determine Whether the Following Two Strings Are Anagrams of Each Other?***

Re-order the strings alphabetically and then compare the two strings.

```
string a = "abcd";
string b = "bcda";

bool AreAnagrams(string inputA, string inputB)
{
   // We reorder the strings alphabetically
   var orderedA = String.Concat(inputA.ToLower().OrderBy(c
=> c));
   var orderedB = String.Concat(inputB.ToLower().OrderBy(c
=> c));

   // Now if they contain the same letters, they should be the
exact same string
   return orderedA == orderedB;
}
```

```
Console.WriteLine(AreAnagrams(a, b));
```

Array

The next section covers common questions related to arrays.

***How Do You Find Duplicate Numbers in an Array With Multiple Duplicates?***

1. Create a dictionary where the number will be the key, and then set the number of times it appears as the value.
2. Loop through the array and increment the count for the key entry in the dictionary.
3. Finally, loop through all pairs in the dictionary; if the value is more than 1, it is duplicated in the array.

```
int[] array = { 5, 2, 1, 5, 8, 5, 1, 7, 7, 0 };
var duplicateDict = new Dictionary<int, int>();

foreach(var val in array)
{
   duplicateDict.TryGetValue(val, out var count);
   duplicateDict[val] = count + 1;
}

foreach (var pair in duplicateDict)
{
   if (pair.Value > 1)
   {
       Console.WriteLine("Value {0} was duplicated and occurred
{1} times.", pair.Key, pair.Value);
   }
}
```

***How Do You Find the Largest and Smallest Number in an Array of 1–100?***

1. Generate an array of all integers from 1 to 100.

2. Have a variable for the largest and smallest numbers you have found so far, initialized to the smallest and largest integer, respectively.
3. Loop through each value in the array and use Max and Min to assign if it is larger or smaller than the numbers you already have stored.
4. These values contain the largest and smallest numbers in the array.

```
// Generate an array of ints containing the numbers from 1 to
100
var array = new int[100];
for (int i = 0; i < 100; i++)
   array[i] = i + 1;

int largestNumber = Int32.MinValue;
int smallestNumber = Int32.MaxValue;;

foreach (var number in array)
{
   largestNumber = Math.Max(largestNumber, number);
   smallestNumber = Math.Min(smallestNumber, number);
}

Console.WriteLine($"Largest Number: {largestNumber},
Smallest Number: {smallestNumber}");
```

***How Do You Remove Duplicates From an Array in Place?***

"In Place" normally means without allocating any further memory. In this case, it is literally impossible to remove something "In Place," as removing this would reduce the array size. Here's how you can do this without a secondary array:

1. Sort the array by ascending value order. This will place duplicates next to each other.
2. Loop through the array from back to front. This is necessary to allow you to reduce it in size while iterating.
3. Track the current value until you find a different one while keeping track of the count of the same value up to this point.

4. If you find a new value or reach the start of the array, use the count to remove the duplicates using swap-back and resize. To do this, swap the unwanted value with the last value in the array and then resize the array down by 1.
5. All duplicates have been removed, but it is no longer a sorted array.

```
int[] array = { 5, 2, 1, 5, 8, 5, 1, 7, 7, 0 };

// First sort the array
Array.Sort(array);

int current = array[^1];
int duplicateCount = -1;

for(var i = array.Length-1; i >= 0; i--)
{
  if (array[i] != current || i == 0)
  {
     // We found a new value or are at the end, so remove
duplicates up to this point
     int startIndex = i + 1;
     for (int j = startIndex; j < startIndex+duplicateCount;
j++)
     {
        array[j] ^= array[^1];
        array[^1] ^= array[j];
        array[j] ^= array[^1];

        Array.Resize(ref array, array.Length - 1);
     }

     // Reset count and current
     current = array[i];
     duplicateCount = 0;
     continue;
  }

  duplicateCount++;
}
```

```
foreach(var val in array)
   Console.Write($"{val} ");
```

***How Do You Find All Pairs in an Array of Integers That Add Up to a Specific Sum?***

1. Loop through an array of values.
2. For each value, loop again through all other values.
3. Check if these two values add up to the provided sum and print them out if so.

```
int[] array = { 5, 2, 1, 5, 8, 5, 1, 7, 7, 0 };

void PrintPairs(int[] input, int sum)
{
   for (int i = 0; i < input.Length; i++)
   for (int j = i + 1; j < input.Length; j++)
       if ((input[i] + input[j]) == sum)
           Console.WriteLine($"Pair
{input[i]}+{input[j]}=={sum}");
}

PrintPairs(array, 10);
```

Binary Tree

This section will cover data structure interview questions on binary trees.

***How Do You Implement a Binary Search Tree?***

A binary search tree is a tree data structure of nodes where each node contains a maximum of two linking nodes, usually known as Left and Right. In its simplest form, a binary search tree can therefore be implemented entirely as a single Node class containing the data and references to the Left and Right nodes.

```
public class Node
{
   public int data;
   public Node left, right;
```

```
  public Node(int item)
  {
     data = item;
     left = right = null;
  }
}

var root = new Node(10);
root.left = new Node(8);
root.right = new Node(2);
```

***How Do You Traverse a Given Binary Tree in Preorder Without Recursion?***

Assuming you already have a binary tree implementation, as described above, follow these steps:

1. Use a stack to process depth-first.
2. Push the first node (usually the root) to the stack.
3. Use a while loop to process so long as the stack is not empty.
   1. Read the top node of the stack.
   2. Remove this from the stack.
   3. If this node has a Left or Right, add them to the stack.
4. The loop will keep adding nodes to the stack and processing them until the whole tree is processed, then the stack will be empty, and the loop will exit.

```
void IterativePreorder(Node node)
{
  Stack<Node> nodeStack = new Stack<Node>();
  nodeStack.Push(node);

  while (nodeStack.Count > 0)
  {
     var topNode = nodeStack.Peek();
     Console.Write(topNode.data + " ");
     nodeStack.Pop();

     if (topNode.right != null)
```

```
            nodeStack.Push(topNode.right);
        if (topNode.left != null)
            nodeStack.Push(topNode.left);
    }
}
```

***Can You Count How Many Leaf Nodes Are in a Given Binary Tree?***

A leaf node is simply a node that has no child nodes. You can use recursion to process the whole tree, and where a node has no Left or Right child nodes, you will add 1 to the leaf count. This will give you the total leaf count in the tree.

```
int CalcLeafCount(Node node)
{
    if (node == null)
        return 0;

    if (node.left == null && node.right == null)
        return 1;

    return CalcLeafCount(node.left) +
CalcLeafCount(node.right);
}
```

***Write a Code To Find the Maximum Depth of a Binary Tree.***

You can solve this with a code using a recursive function to process the whole tree, returning the calculated depth up the recursive call stack.

```
int CalcMaxDepth(Node node)
{
    if (node == null)
        return -1;

    int leftDepth = CalcMaxDepth(node.left);
    int rightDepth = CalcMaxDepth(node.right);

    if (leftDepth > rightDepth)
        return leftDepth + 1;
```

```
   return rightDepth + 1;
}
```

Linked List

These next questions cover the linked list data structure.

***How Do You Reverse a Linked List?***
A linked list is comprised of nodes pointing to the previous and next nodes, so you can reverse a linked list by iterating the nodes in the list, and then swap the previous and next node entries.

```
void ReverseListRecursive(LinkedList linkedList)
{
   linkedList.head = ReverseListRecursiveImpl(linkedList.head);
}

LinkedList.Node ReverseListRecursiveImpl(LinkedList.Node
head)
{
   if(head == null)
      return head;

   if(head.next == null)
      return head;

   LinkedList.Node newHead =
ReverseListRecursiveImpl(head.next);

   head.next.next = head;
   head.next = null;

   return newHead;
}
```

***How Do You Reverse a Singly Linked List Without Recursion?***

The principle of reversing a linked list without recursion is the same, and is in fact much simpler than the recursive solution. There is almost never a scenario where you should use recursion to iterate a linked list.

```
void ReverseList(LinkedList linkedList)
{
   LinkedList.Node prev = null;
   var current = linkedList.head;

   while (current != null)
   {
       var next = current.next;
       current.next = prev;
       prev = current;
       current = next;
   }

   linkedList.head = prev;
}
```

***How Do You Determine Whether There Is a Loop in a Singly Linked List?***

A loop (or cycle) in a linked list is when a node’s next value points back to an earlier node in the list.

1. Create a hash set of nodes. You will use this to track whether or not you have already processed a node.
2. Iterate the linked list, adding each node to the hash set.
3. If the node is already present in the hash set, then the linked list contains a loop (or cycle).

```
bool ContainsLoop(LinkedList linkedList)
{
   HashSet<LinkedList.Node> nodeVisitSet = new
HashSet<LinkedList.Node>();

   var node = linkedList.head;
   while (node != null)
```

```
  {
     // If it's there, we already visited so this is a loop
     if (nodeVisitSet.Contains(node))
        return true;

     nodeVisitSet.Add(node);

     node = node.next;
  }

  return false;
}
```

***How Do You Add an Element to the Middle of a Linked List?***

You can insert an element in the middle of a linked list by modifying the middle element to point to the new node. Next, modify the new node to point to the next node (which the middle node had previously pointed to).

```
void Insert(LinkedList.Node insertPos, LinkedList.Node
newNode)
{
  var oldNext = insertPos.next;
  insertPos.next = newNode;
  newNode.next = oldNext;
}
```

Other

The following section includes algorithm questions, Armstrong numbers, and Fibonacci numbers.

***How Is a Bubble Sort Algorithm Implemented?***

A bubble sort algorithm is implemented by iterating an array and repeatedly swapping pairs that are in the wrong order.

```
void BubbleSortArray(int[] array)
{
  int n = array.Length;
  for (int i = 0; i < n - 1; i++)
  for (int j = 0; j < n - i - 1; j++)
  {
     if (array[j] > array[j + 1])
     {
        (array[j], array[j + 1]) = (array[j + 1], array[j]);
     }
  }
}
```

***Write a Program To Find the Prime Factors of an Integer.***

1. First, as long as the number is divisible by 2, print out 2 and divide it by 2.
2. Now that you have an odd number, loop from 3 to the square root of the number, incrementing by 2 each time.
3. While the number is divisible by i, print it out and divide by i.
4. Lastly, if the remaining number is larger than 2, print it out.

```
void PrintPrimeFactors(int number)
{
  while (number % 2 == 0)
  {
     Console.Write(2 + " ");
     number /= 2;
  }

  for (var i = 3; i <= Math.Sqrt(number); i+= 2)
  {
     while (number % i == 0)
     {
        Console.Write(i + " ");
        number /= i;
     }
  }

  if (number > 2)
     Console.Write(number);
}
```

***Check if the Following Number Is an Armstrong Number.***
An Armstrong number is when the sum of the digits raised to the power of the number of digits equals the number itself. For example, 371 == 3^3 + 7^3 + 1^3.

```
bool IsArmstrongNumber(int number)
{
   // Count the number of digits
   int count = 0;
   int numberCursor = number;
   while (numberCursor > 0)
   {
       numberCursor /= 10;
       count++;
   }

   // For each digit, raise it to the power of the digit count
   // and sum them together
   int sum = 0;
   numberCursor = number;
   while (numberCursor > 0)
   {
       sum += (int)Math.Pow(numberCursor % 10, count);
       numberCursor /= 10;
   }

   // Check if the sum is the same as the number
   return sum == number;
}
```

***Write a Program To Print the First N Fibonacci Numbers.***
Fibonacci numbers are numbers that equal the sum of the preceding two numbers. This is using an iterative method starting from 0 to N. For each Fibonacci number, add the previous two numbers to get the next number in the set.

```
void PrintFibonacci(int n)
{
   var a = 0;
   var b = 1;
```

```
    for (var i = 0; i < n; i++)
    {
        var temp = a;
        a = b;
        b = temp + b;

        Console.WriteLine(a);
    }
}

PrintFibonacci(7);
```

***How Do You Implement a Bucket Sort Algorithm?***

1. Create a number of “buckets” for the length of the input array. Each bucket is a list of entries.
2. Iterate the input array, storing each value in a computed bucket index. The computed bucket index is the value of the current input array entry multiplied by the number of buckets.
3. Sort each bucket’s list.
4. Iterate all values in all buckets and write them back into the input array in this order.

```
void BucketSort(float[] array)
{
    var numBuckets = array.Length;
    var buckets = new List<float>[numBuckets];

    for (var i = 0; i < numBuckets; i++)
        buckets[i] = new List<float>();

    for (var i = 0; i < numBuckets; i++)
    {
        var bucketIndex = array[i] * numBuckets;
        buckets[(int)bucketIndex].Add(array[i]);
    }

    for (var i = 0; i < numBuckets; i++)
        buckets[i].Sort();
```

```
  var arrayIndex = 0;
  for (var i = 0; i < numBuckets; i++)
  {
     for (var j = 0; j < buckets[i].Count; j++)
     {
        array[arrayIndex++] = buckets[i][j];
     }
  }
}
```

**Coding Interview Questions [Concept-Based]**
Concept-based questions evaluate your understanding of code-related concepts, as well as your ability to articulate this understanding.

What Is an Array?

An array data structure is a group of similar data elements stored at adjacent memory locations, such as an integer array.

What Is a Stack?

A stack is an abstract data element and a collection of data elements. Elements can be added to the stack with the operation "push," and removed with "pop."

What Is a Matrix?

Matrix" is a term used for a table of numbers made up of rows and columns.

What Is a Tree?

A tree is a non-linear data structure that allows for multiple relations between nodes.

What Is a Heap?

A tree-based structure that works as a complete binary tree. There are two types of heaps: a min-heap and a max-heap. The root node of a min-heap must be the lowest value of all the nodes, and the root node of a max-heap must be the highest.

What Is a Queue?

A queue is a collection of entities that are kept in a specific sequence. Entities can be added to one end and removed from the other.

What Is a Graph?

A graph is a non-linear data structure made up of nodes and edges. The edges can connect any two nodes in the graph.

What Is a Linked List?

A linked list is a linear data structure where the entities are not stored next to each other on the memory. Instead, each node includes a reference to the following node in the list.

What Is a Hash Table?

Source: Khalil Stemmler
A hash table is a data structure that stores information through association. Each data element has a unique index value that identifies it and allows it to be accessed quickly.

What Is a Binary Search Tree?

A binary search tree uses the tree structure mentioned above. Every value in the left subtree is greater than the value of the root node, and every value in the right subtree is lower.

What Is Recursion?

Recursion is a way to solve problems that rely on looping code. Here’s how you do that:

1. Tell the computer to stop when the goal is reached.
2. Tell the computer to take a very simple step towards the goal.
3. Order the algorithm to be repeated.

This way, the code can check whether the criteria defined have been met and if they haven’t, proceed to take a small step forward. The algorithm will then repeat until the criteria are all met and the second step is not required.

## What Is a Sorting Algorithm?

A sorting algorithm uses a set collection of instructions to sort the items of an array or list into a certain order. This is most commonly a numerical or alphabetical order.

## What Is a Searching Algorithm?

A set of instructions that confirms the existence of or retrieves certain data within a larger group of data.

## What Is a Hashing Algorithm?

A hashing algorithm assigns a fixed value to a data element, allowing it to be used in a hash table.

## What Is a Greedy Algorithm?

A greedy algorithm approaches each stage of a problem in isolation, selecting the most optimal solution for that particular stage without considering any other factors.

## What Is the “Big O” Notation?

he “Big O” is a mathematical notation that measures the efficiency of an algorithm by conserving the amount of time it takes and the amount of space it uses. It always measures based on the worst-case scenario.

## Do You Think OOP Is a Good Coding Method?

Object-oriented programming (OOP) has been the dominant paradigm for decades, though it is debated whether this was due to merit or chance. In more recent years, data-oriented programming (DOP) has also gained popularity and can be used to great effect in many areas. However, rather than saying one paradigm is better than the other, it is more accurate to say that there are pros and cons to both.

## What Are Leaf Nodes and Why Are They Important in a Binary Tree?

Nodes in a binary tree that have children are referred to as interior nodes. When a node has no children (and therefore is a dead end), it’s called a leaf node. A leaf node signifies the end of a branch and lets the computer know that it doesn’t need to be processed any further.

## What Is the Difference Between a Programming Language and a Script?

The main difference between the two is that a programming language is compiled before running, meaning it is converted to machine code that the computer can directly read. Scripting languages, however, are instead interpreted by the computer in real-time.

## What Is the Difference Between a Class and an Object?

A class defines the properties of an object, and every object fits within a class. In other words, classes are categories, and objects are sorted into categories.

What Is the Difference Between an Interpreter and a Compiler?

A compiler converts an entire program into machine code before running it, whereas an interpreter changes each line one by one as it runs the program.

What Is the Difference Between an Array and a Linked List?

Both are collections of similar data types, but they differ in the way the data is stored. An array stores data in adjacent locations in the memory, whereas each element in a linked list includes a reference to the following element.

What Is the Difference Between a Stack and a Queue?

Both a stack and a queue are ordered lists of data, but while queues can have elements added or removed from both ends, a stack only allows it to happen from one end.

What Is the Difference Between a String and a StringBuilder in Java?

A string in Java is known as “immutable.” This means its contents cannot be altered after it has been created. A StringBuilder, however, can be changed.

What Is the Difference Between a Breadth-First and a Depth-First Search?

When searching a binary tree, the depth-first approach will follow the nodes downward and process them in that order. In comparison, a breadth-first search will process all adjacent nodes before moving downward.

What’s the Difference Between Searching and Sorting?

Searching is the act of checking for and retrieving a specific data element, whereas sorting rearranges multiple data elements in an array or list into some kind of order.

What Is the Difference Between a Static and a Dynamic Programming Language?

Static language:

- The coder must declare the data types of variables before they can be assigned a value.
- The computer will check these data types when the code is compiled, and any errors must be fixed before the code can be run.

Dynamic language:

- Data types do not need to be declared by the coder.
- The data types will be checked at runtime, meaning that code containing errors will attempt to run.

**Coding Interview Questions & Answers [General]**
Here are some of the general questions you might face, pertaining to your interests, motivations, and aspirations related to coding.

Tell Me About Yourself

**Your interview is an opportunity to stand out from the competition. Impressing recruiters with your skills is one way to do this, but you can also do this by telling a compelling story about yourself and your coding career**
Why Did You Opt for a Career in Coding?

This question helps interviewers judge your character and your professional motivations, but you don't need to have an impressive tale to tell. Think about what got you into coding: was it a person, a passion, or an aspiration to interview at a top tech company?

What Is Your Favorite Programming Language?

It's likely that your dominant programming language is your favorite since you've spent so much time with it. This is also a good answer to give because it shows that you have a passion for the language you'll be working with. You can, however, also give an honorable mention to another language.

## What Is the Most Difficult Project You've Encountered on Your Learning Journey?

Not everyone finds the same things difficult, and there is no right or wrong when it comes to your own personal strengths and weaknesses. With this in mind, it's perfectly acceptable to be honest here, and it's much more likely to produce an interesting answer than a template answer.

## What Have You Done To Improve Your Skills as a Coder?

In this question, the interviewer wants to know how self-driven, motivated and passionate you are about your career. How have you gone above and beyond to improve your skills as a coder?

This is a great time to talk about any certifications you have, online bootcamps you've completed, or interesting coding projects you have in your personal portfolio.

## What's Your Key Strength?

All aspects of coding are important, and companies will ideally want an expert in every area. You might be good at debugging, liaising with non-technical colleagues, or coming up with out-of-the-box solutions. Whatever your skill, all you need to do is sell how it could be useful to the company.

## What Is Your Greatest Weakness?

For this question, it's best to avoid non-answers like being too much of a perfectionist or working too hard. Be sincere—what do you struggle with, and what do you do to combat it and try to improve?

## Why Do You Want To Work for This Company?

Answering this question requires prior research of the company you are interviewing with. Sometimes it is necessary to send out applications in such a large number that you can’t really focus on the individual companies. Even so, if you progress to the interview stage, you need to make sure you know the company and why you are interested in it.

## Situational Questions Based on Your Resume

These questions will focus specifically on the prior experience you have listed on your resume. The interviewer will use them to get an idea of how you deal with various challenges in the workplace.

### ***Can You Explain These Gaps in Your Resume?***

There are many different reasons you might have gaps between jobs listed on your resume, and sometimes you may be asked about them. It’s a tricky question, but the interviewer is often more interested in how you answer rather than the actual reason for the gaps.

The key is to be confident, unapologetic, and clear in your answer while keeping it brief and to the point. The reason could be quite personal, but your answer shouldn’t be your life story.

### ***You Have Multiple Programming Languages Listed on Your Resume. Could You Tell Me Which Ones You Have the Most Experience With and Where You Encountered Them?***

This is a common question during a programming interview, as many programmers will list multiple languages on their resumes. While it can be tempting to list as many as you can, it’s important to make it clear which are your dominant languages, and which you have used in a professional environment.

When answering this question, it can be helpful to practice quality over quantity. Programmers are often hired to work in one main language, so demonstrating the depth of your knowledge in your dominant language is more relevant than an overview of all the languages you've had experience with.

## FAQs About Coding Interview Questions

Finally, let's look at some frequently asked questions on coding interview questions.

How Long Do Coding Interviews Generally Last?

Most coding interviews are 45-60 minutes long.

Are Coding Interviews Hard?

Each company will have its own style of interviewing, and some will include more technical questions than others. However, if you believe you have the knowledge to do the job, you will have what it takes to pass the interview too, and it should not be too hard.

How Should You Prepare for a Coding Interview?

Some standard interview questions, such as "how can you reverse a string?" will cover concepts you potentially haven't used in a long time, so brushing up on these areas can be a good idea.

It's also good practice to plan answers to general questions about yourself, research the company, and plan questions to ask your interviewer.

What Are Some Good Questions To Ask the Interviewer?

Companies are always looking for talented individuals who will stay with the company for a long time and provide as much value as possible. Some good questions to show your potential in this area would be:

1. Where do you see the company in five years?
2. What opportunities are there for training and progression within the company?
3. What would my daily responsibilities look like?

List of 20 C Coding Interview Questions and Answer Here is a list of 20 C coding interview questions and answers: 1. Find the largest number among the three numbers. C // C Program to find // Largest of three numbers
#include<stdio.h>
 int main()
{ int a = 1, b = 2, c = 3; // condition for a is greatest if (a > b && a > c) printf("%d", a); // condition for b is greatest else if (b > a && b > c) printf("%d", b); // remaining conditions // c is greatest else printf("%d", c); return 0; } Output 3 2. Write a Program to check whether a number is prime or not. // C Program for // Checking value is // Prime or not #include int main() {int N = 91; int flag = 0; // Iterate from 2 to N/2 for (int i = 2; i <= N / 2; i++) { // If n is divisible by any number between 2 and // n/2, it is not prime if (N % i == 0) { flag = 1; break; } } if (flag == 0) printf("Not a Prime Number"); else printf("Is a Prime Number"); return 0; } Output Is a Prime Number
3. Write a C program to calculate Compound Interest. // C program to calculate Compound Interest #include // For using pow function we must

// include math.h #include // Driver code int main() { // Principal amount double principal = 2300; // Annual rate of interest double rate = 7; // Time double time = 4; // Calculating compound Interest double amount = principal * ((pow((1 + rate / 100), time))); double CI = amount - principal; printf("Compound Interest is : %lf", CI); return 0; } Output Compound Interest is : 714.830823

4. Write a Program in C to Swap the values of two variables without using any extra variable. // C Program to

// Swap two numbers // No Extra Space #include int main() { int x = 10; int y = 20; printf("x: %d , y: %d\n", x, y); // Code to swap 'x' and 'y' x = x + y; y = x - y; x = x - y; printf("x: %d , y: %d\n", x, y); return 0; } Output x: 10 , y: 20 x: 20 , y: 10 5. Write a Program to Replace all 0’s with 1’s in a Number. // C Program for // Replacing 0 to 1 #include

#include int main() { int N = 102301; int ans = 0; int i = 0; while (N != 0) { // Condition to change value if (N % 10 == 0) ans = ans + 1 * pow(10, i); else ans = ans + (N % 10) * pow(10, i); N = N / 10; i++; } printf("%d", ans); return 0; } Output: 112311
6. Write a Program to convert the binary number into a decimal number. // C Program for converting

// binary to decimal #include int main() { int N = 11011; // Initializing base value a to 1 int a = 1; int ans = 0; while (N != 0) { ans = ans + (N % 10) * a; N = N / 10; a = a * 2; } printf("%d", ans); return 0; } Output

7.  Write a Program to check if the year is a leap year or not. // C Program to check // Year is leap year or not #include // Function Declaration to check leap year
void leap_year(int year) { // If a year is multiple of 400, then leap year if (year % 400 == 0) printf("%d is a leap year.\n", year); // If a year is multiple of 100, then not a leap year else if (year % 100 == 0) printf("%d is not a leap year.\n", year); // If a year is multiple of 4, then leap year else if (year % 4 == 0) printf("%d is a leap year.\n", year); // Not leap year else printf("%d is not a leap year.\n", year); } int main() { leap_year(2000); leap_year(2002); leap_year(2008); return 0; } Output 2000 is a leap year. 2002 is not a leap year.
2008 is a leap year
8. Write a program to Factorial of a Number. // C Program to calculate // Factorial of a number #include // Calculating factorial using iteration void factorial_iteration(int N) { unsigned long long int ans = 1; for (int i = 1; i <= N; i++) { ans = ans * i; } printf("Factorial of %d is %lld\n", N, ans); } //

```
Calculating factorial using recursion int factorial(int N) { if (N
== 0) return 1; // Recursive call return N * factorial(N - 1); }
int main() {
int n; n = 13; factorial_iteration(n); n = 9; printf("Factorial of
%d using recursion:%d\n", n, factorial(n)); return 0; } Output
Factorial of 13 is 6227020800 Factorial of 9 using
recursion:362880
```