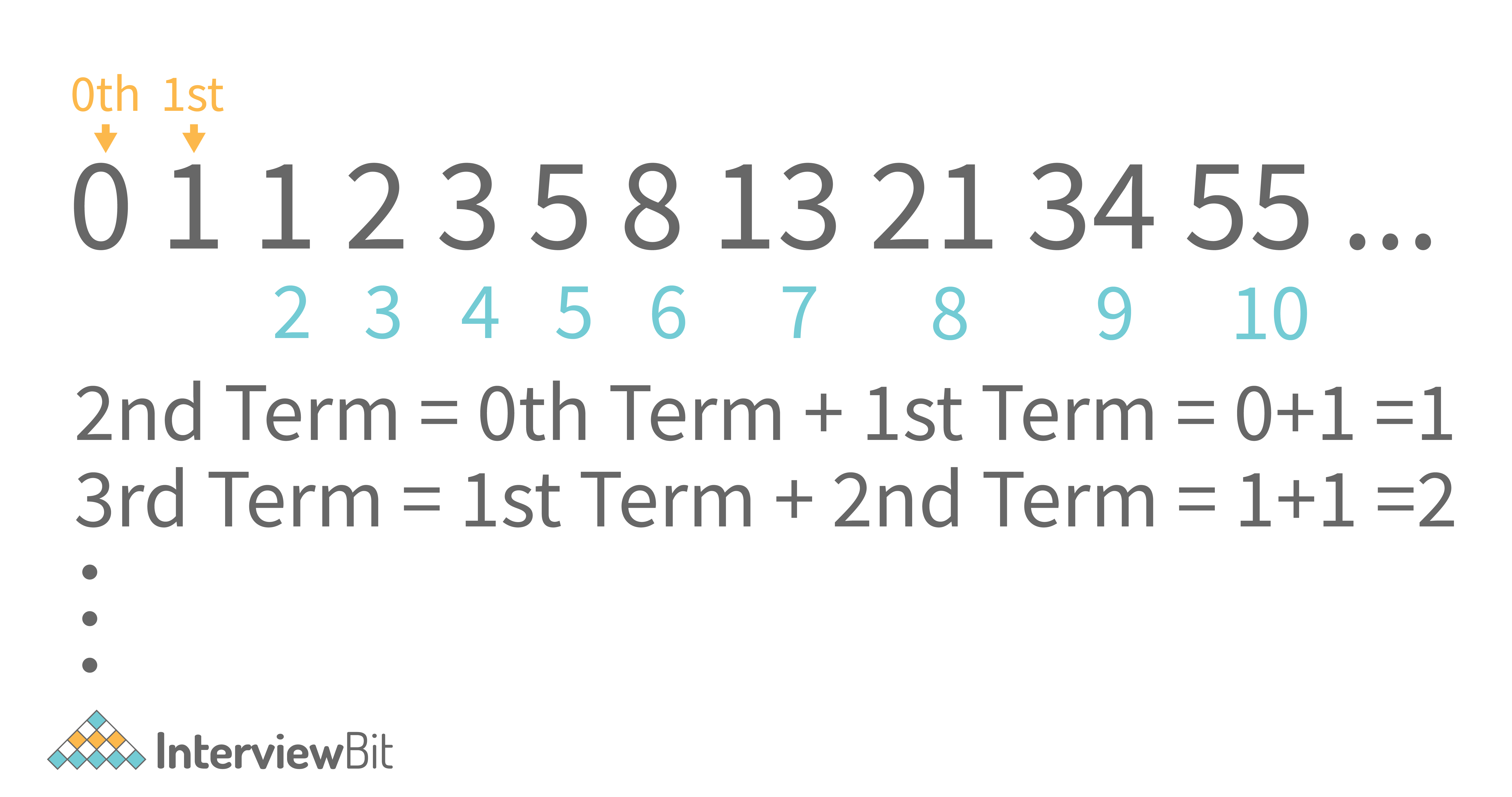
**Java Programming Interview Questions for Freshers**

**1. Write a program in Java to generate the Nth Fibonacci Number using Iteration and Constant Space.**

Fibonacci Series is a series in which the Nth term is the sum of the previous 2 terms i.e. (N-1)th and (N-2)th terms. The first 2 terms of the Fibonacci sequence are always known. They are 0 and 1 respectively. After this, the further terms can be generated as follows:



So, in general, we can derive a generic term i.e.

Fib(N) = Fib(N - 1) + Fib(N - 2)

So, let us now write a program to find the Nth Fibonacci Number using iteration.

**a. Java Program to generate Nth Fibonacci Number using Iteration.**

**import** java.util.\*;

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**int** n = scn.nextInt();

**int** a = 0; //0th fibonacci number

**int** b = 1; //1st fibonacci number

**if**(n < 0) {

System.out.println("N cannot be negative");

**return**;

}

**if**(n == 0) System.out.println(a);

**else** **if**(n == 1) System.out.println(b);

**else** {

**int** c = 0;

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

c = a + b;

a = b;

b = c;

}

System.out.println(c);

}

}

}

**Sample Input/Output:**[**Run Code**](https://www.interviewbit.com/snippet/bc04f255bbf409133731/)

Input: 7

Output: 13

* **Corner Cases You might Miss:**The simple mistake of not handling the corner case when N is negative can happen to a lot of programmers. Since the number of terms can’t be negative, this should be handled separately as shown in the code above.
* **Time Complexity:**O(N) because we have to travel N terms
* **Auxiliary Space:**O(1) as no extra space is used.
* **Follow up:**You can also solve this problem using dynamic programming. This will take up O(N) space as well and the time complexity will be the same i.e. O(N). Try the dynamic programming approach yourself.

**Create A Free Personalised Study Plan**

Get into your dream companies with expert guidance

Real-Life Problems

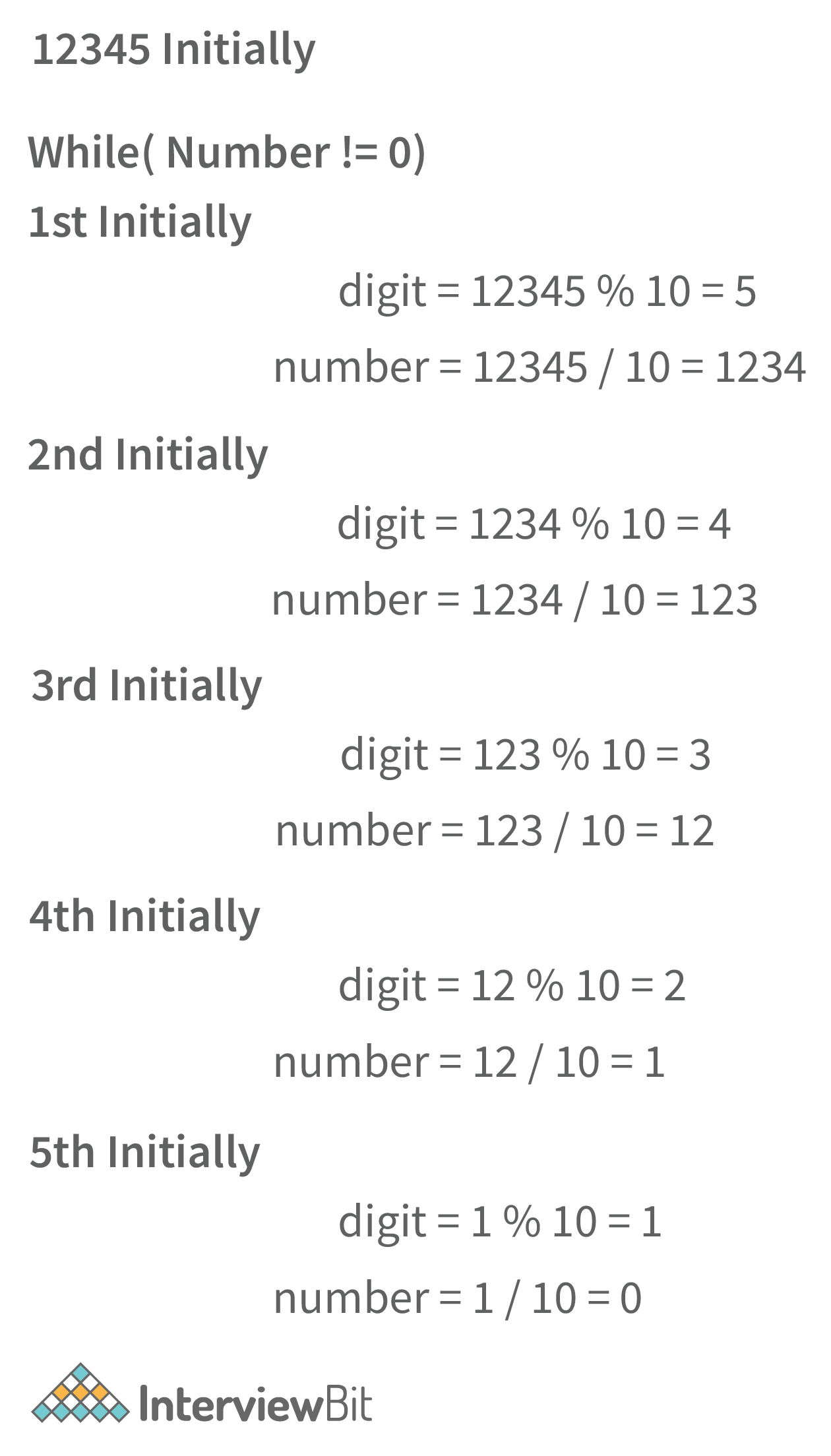
Prep for Target Roles

Custom Plan Duration

[**Create My Plan**](https://www.interviewbit.com/interview-preparation-kit/)

**2. Write a program in Java to count the digits in a number.**

Let us consider the number 12345. This is a 5 digit number. The number of digits can be counted as follows:



In the image above, we have shown the way of extracting the digits of a number. However, in our questions, we just need to count the digits in the number. So, we have to count the number of times we can divide our input number by 10 before it becomes 0.

Let us write the code based on the above algorithm.

**Java Program to Count the Number of Digits in a Number.**

**import** java.util.\*;

**class** **Main** {

**public** **static** **int** **countDigits**(**int** n) {

**if**(n == 0) **return** 1;

//if a negative number is entered

**if**(n < 0) n = -n;

**int** res = 0;

**while**(n != 0) {

n = n/10;

res++;

}

**return** res;

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**int** n = scn.nextInt(); //input the number

System.out.println("The number of digits in " + n + " are: " + countDigits(n));

}

}

**Output**

* For Positive Number

Input: 1234

Output: The number of digits in 1234 is: 4

* For 0

Input: 0

Output: The number of digits in 0 is: 1

* For Negative Number

Input: -12345

Output: The number of digits in -12345 is: 5

* **Corner Cases You Might Miss:**We have used the loop and carried on iterations till the number becomes 0. What if the number was already 0? It still has 1 digit. So, we have handled that separately. Also, to avoid any confusion, the negative numbers are converted to positive in our function and then we calculate their number of digits.
* **Time Complexity:**O(log10N) where N is the input number. This is because we keep dividing the number by 10.
* **Auxiliary Space:**O(1) as we have not used any extra space.

**3. Write a program in Java to calculate the number of times a digit ‘D’ appears in a number N. You have to take N and D as inputs from the user.**

This is the follow-up question to the previous question. In the previous question, we discussed how you can check the value of a digit using the modulus (%) operator. So, we will just use the previous code and in every iteration, we will check whether the digit is “D” or not. If it is D, increment the counter. The program for the same is shown below:

**Java Code for Calculating Frequency of a Digit D in a Number N**

**import** java.util.\*;

**class** **Main** {

**public** **static** **int** **countDigitFreq**(**int** n,**int** D) {

**if**(n == 0 && D == 0) **return** 1; //number 0 has 1 frequency of 0

//if a negative number is entered

**if**(n < 0) n = -n;

**int** counter = 0;

**while**(n != 0) {

**int** digit = n % 10; //calculate the digit

**if**(digit == D) counter++;

n = n/10;

}

**return** counter;

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**int** n = scn.nextInt(); //input the number

**int** d = scn.nextInt(); //input the digit

**int** x = countDigitFreq(n,d);

System.out.println("The digit " + d + " occurs " + x + " times in " + n);

}

}

**Sample Input/Output**

Input: 142454

Output: The digit 4 occurs 3 times in 142454

* **Corner Cases You Might Miss:**If the input number is 0 and the digit is also 0, it becomes a crucial corner case. This is because the number 0 has 1 frequency of digit 0 but it will not be handled correctly by our loop. So, we do this separately. Also, we have converted the negative numbers to positive ones to solve this problem.
* **Time Complexity:**O(log10N) where N is the input number. This is because we keep dividing the number by 10.
* **Auxiliary Space:**We have not used any auxiliary Space here. So, it is O(1).

**You can download a PDF version of Java Programming Interview Questions.**

[**Download PDF**](javascript:void(0))

**4. Write a Java Program to calculate xn (x to the power n) using Recursion. You can use O(N) time but can’t use any extra space apart from the Recursion Call Stack Space.**

In a recursive function, we need to find a relation between the smaller problem and the larger problem. Here, we have a clear mathematical relationship between the large problem xn and the problem that is smaller than it i.e. x to the power (n-1). The relation is:

�� = � \* ��-1

The relation is: **(x to the power n)  = x \* x to the power(n-1)**

In terms of programming (using functions), we can write this relation as: **power(x,n) = x \* power(x,n-1)**

For example, **2 to the power 5 = 32**. This can be calculated as **2 \* (2 to the power 4) = 2 \* 16 = 32**. So, we have found the recurrence relation in the above equation. Now, what will be the base case?

The base case will be the smallest such problem that we can solve. So, the smallest problem is calculating the power of 0 to any number. **We know that any number to the power 0 gives the result as 1.**So, this will be our **base case**.

Now, let us write the code for the same.

**Java Code to Calculate x to the power n**

**import** java.util.\*;

**class** **Main** {

**public** **static** **double** **power**(**double** x, **int** n) {

**if**(n == 0) **return** 1.0;

**double** xpnm1 = power(x,n-1); //x power n-1 (xpnm1)

**return** x \* xpnm1;

}

**public** **static** **double** **pow**(**double** x, **int** n) {

**if**(n < 0) {

**return** 1.0 / power(x,-n);

}

**return** power(x,n);

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**double** x = scn.nextDouble();

**int** n = scn.nextInt();

System.out.println(pow(x,n));

}

}

**Sample Output:**

* For positive Power

Input:

1.10

3

Output: 1.3676310000000003

* For negative Power

Input:

1.110

-3

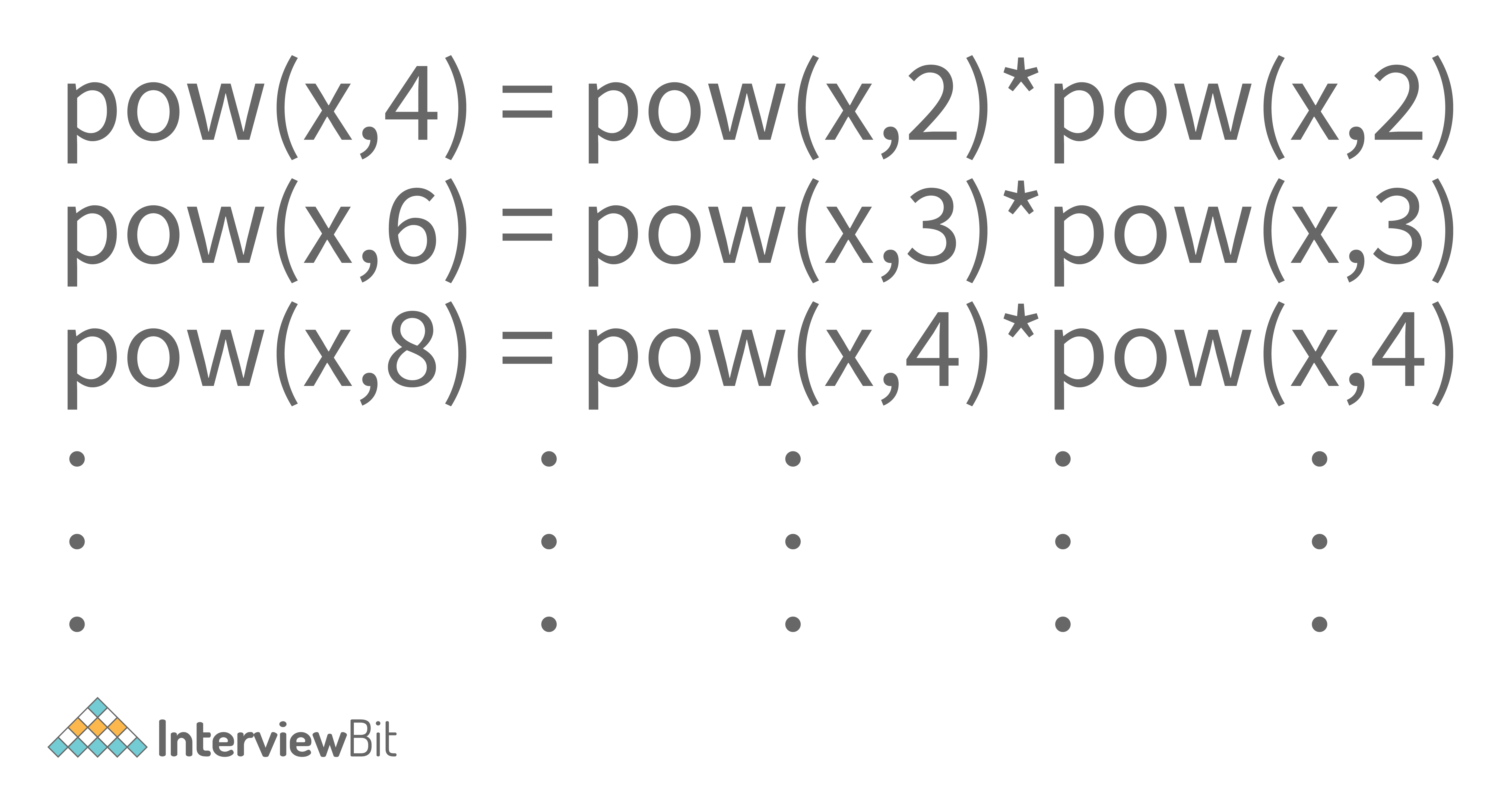
Output:

0.7311913813009502

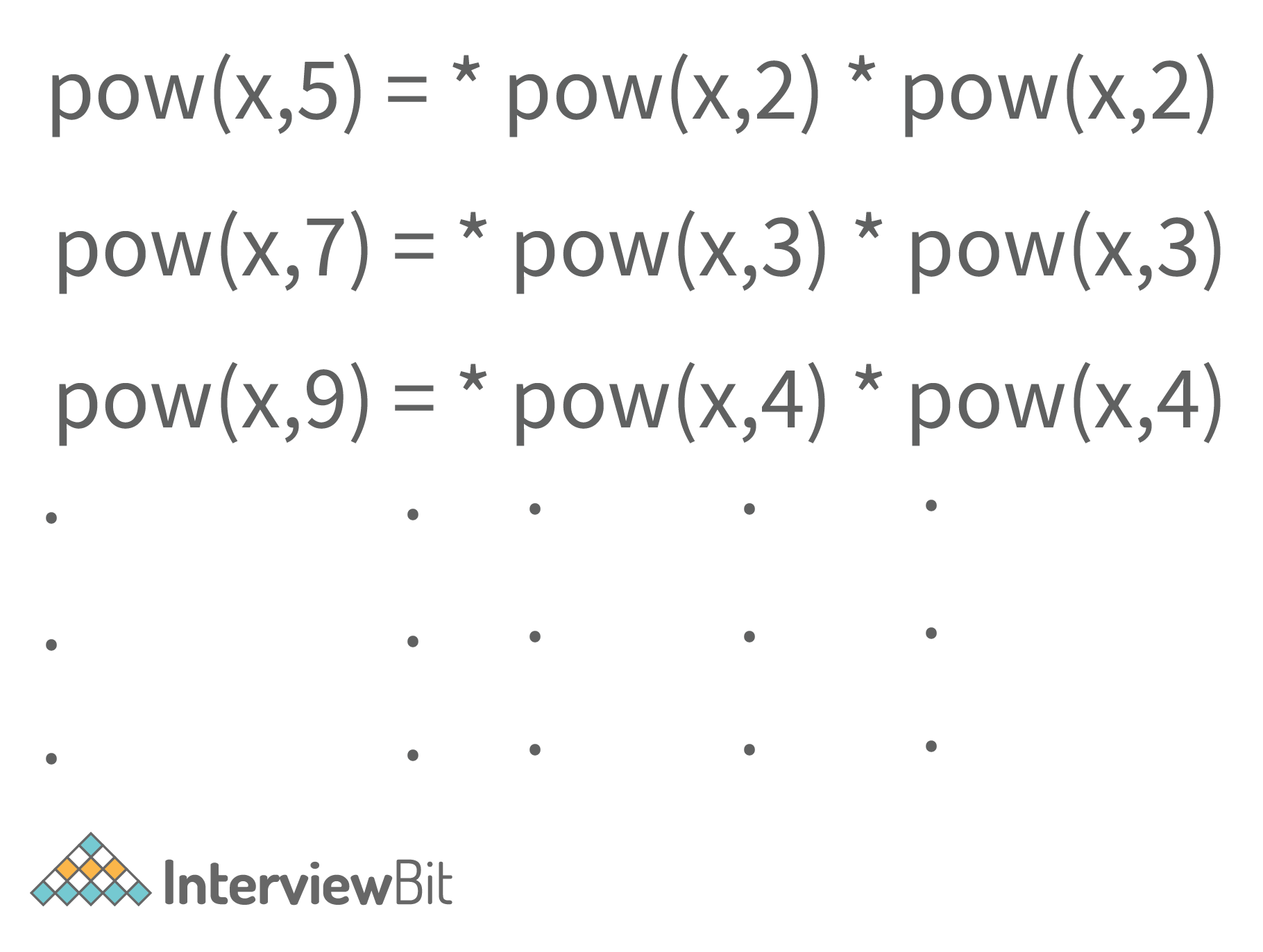
* **Corner Cases You Might Miss:**The power of a number can be negative too. So, we know that (x to the power -n) = [1/(x to the power n)]. In this way, we can handle the corner test case of a power being negative. What if the number is negative? Is our code handling that case? Yes, it does. Why? Try to think about this.
* **Time Complexity:**The time complexity of this code is O(N) where N is the power. We see that the time complexity does not depend on X i.e. the number. It only depends on the power of the number.
* **Auxiliary Space:**The auxiliary space is O(1) as we have not used any extra space.

**5. Write a program in Java to calculate pow(x,n) using recursion. The expected time complexity is O(log2N) where N is the power. You cannot use any extra space apart from the recursion call stack space.**

We have seen how we can calculate the pow(x,n) in linear time. We can optimize our approach by changing the recurrence relation. Let us understand this with the help of an example shown below:

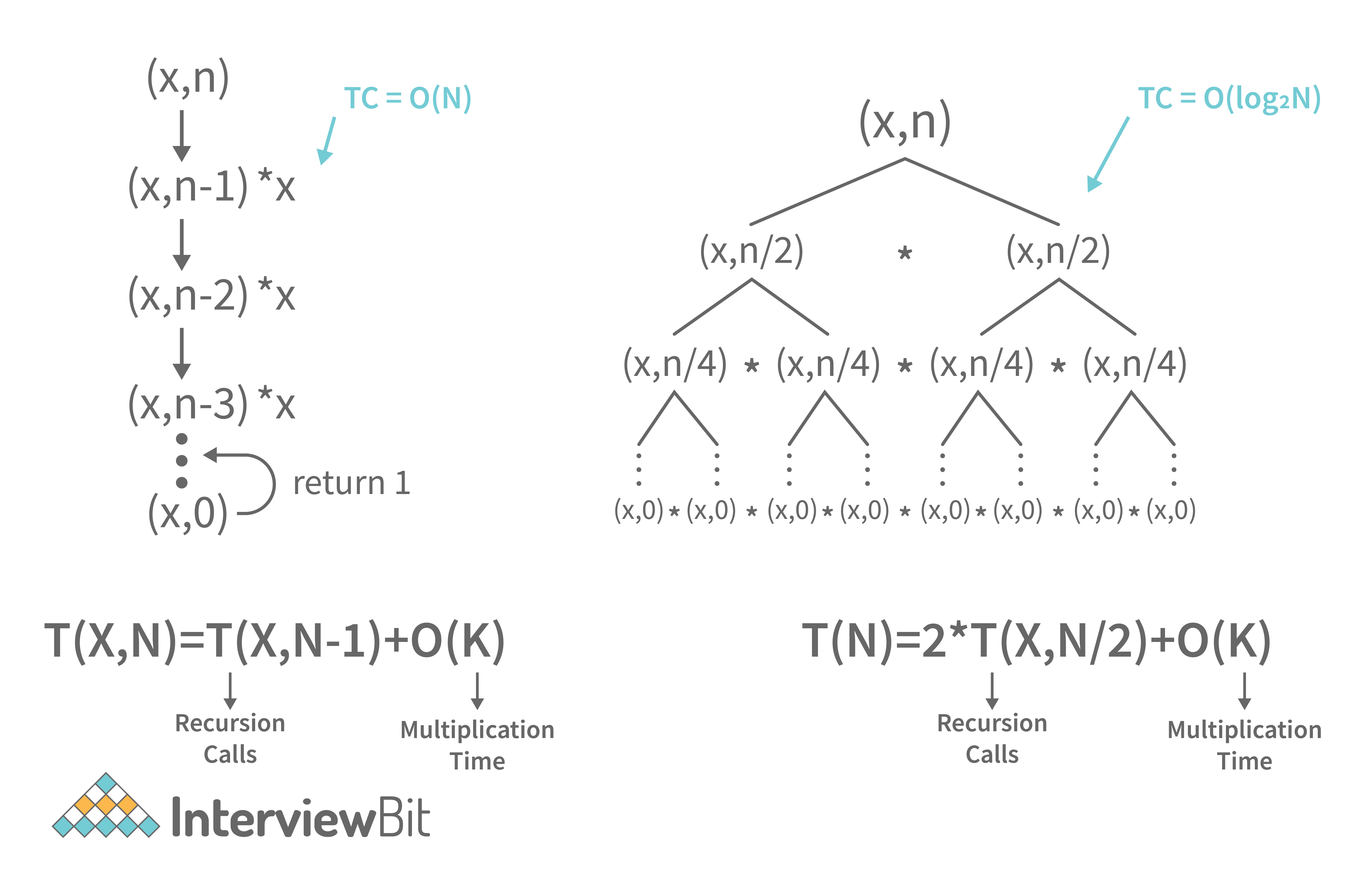


So, we can see that if the power is even, we can divide the power by 2 and can multiply x to the power n/2 by itself to get our answer. What if the power of the number is odd?



In that case, we multiply the number x once to the term x to the power n/2 multiplied by itself. Here, n/2 will be the floor value of (n/2). You can verify this for any pair of x and n.

So, doing this will reduce the time complexity from O(N) to O(log2N). This happens because of the change in recurrence relation and the recursion tree as shown below.



Solving these recurrence relations gives us the respective time complexities. So, the code for O(log2N) approach is shown below

**Java Code for (x to the power N) in Logarithmic Time Complexity**

**import** java.util.\*;

**class** **Main** {

**public** **static** **double** **power**(**double** x, **int** n) {

**if**(n == 0) **return** 1.0;

**double** xpnby2 = power(x,n/2); //xpnby2 = x power n by 2

**if**(n % 2 == 0) **return** xpnby2 \* xpnby2; //if power is even

**return** x \* xpnby2 \* xpnby2; //if power is odd

}

**public** **static** **double** **pow**(**double** x, **int** n) {

**if**(n < 0) {

**return** 1.0 / power(x,-n);

}

**return** power(x,n);

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**double** x = scn.nextDouble();

**int** n = scn.nextInt();

System.out.println(pow(x,n));

}

}

**Sample Output:**

* For positive Power

Input:

1.10

3

Output: 1.3676310000000003

* For negative Power

Input:

1.110

-3

Output:

0.7311913813009502

* **Corner Cases, You Might Miss:**The power of a number can be negative too. So, we know that x-n = (1/xn). In this way, we can handle the corner test case of a power being negative.
* **Time Complexity:**As already discussed, Time Complexity is O(log2N).
* **Auxiliary Space:**The auxiliary space is O(1) as we have not used any extra space.

**Learn via our Video Courses**

[RAHUL JANGHU](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLWZvci1iZWdpbm5lcnM/dXRtX3NvdXJjZT1pYg==" \t "_blank)

**[Python Course for Beginners With Certification: Mastering the Essentials](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLWZvci1iZWdpbm5lcnM/dXRtX3NvdXJjZT1pYg==" \t "_blank)**

[4.90](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLWZvci1iZWdpbm5lcnM/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Enrolled: 101890](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLWZvci1iZWdpbm5lcnM/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLWZvci1iZWdpbm5lcnM/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[MRINAL BHATTACHARYA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvamF2YXNjcmlwdC1iZWdpbm5lcnM/dXRtX3NvdXJjZT1pYg==" \t "_blank)

**[JavaScript Course With Certification: Unlocking the Power of JavaScript](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvamF2YXNjcmlwdC1iZWdpbm5lcnM/dXRtX3NvdXJjZT1pYg==" \t "_blank)**

[4.8](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvamF2YXNjcmlwdC1iZWdpbm5lcnM/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Enrolled: 54089](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvamF2YXNjcmlwdC1iZWdpbm5lcnM/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvamF2YXNjcmlwdC1iZWdpbm5lcnM/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[YASH RAJ](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvYXdzLWZyZWUtY291cnNlP3V0bV9zb3VyY2U9aWI=" \t "_blank)

**[AWS Free Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvYXdzLWZyZWUtY291cnNlP3V0bV9zb3VyY2U9aWI=" \t "_blank)**

[4.7](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvYXdzLWZyZWUtY291cnNlP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Enrolled: 240](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvYXdzLWZyZWUtY291cnNlP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvYXdzLWZyZWUtY291cnNlP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[MRINAL BHATTACHARYA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1yZWFjdC1qcy1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

**[React JS Free Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1yZWFjdC1qcy1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)**

[4.8](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1yZWFjdC1qcy1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Enrolled: 342](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1yZWFjdC1qcy1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1yZWFjdC1qcy1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[SUBHESH KUMAR](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvb2JqZWN0LW9yaWVudGVkLXByb2dyYW1taW5nLWphdmE/dXRtX3NvdXJjZT1pYg==" \t "_blank)

**[Object Oriented Programming in Java Course Online](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvb2JqZWN0LW9yaWVudGVkLXByb2dyYW1taW5nLWphdmE/dXRtX3NvdXJjZT1pYg==" \t "_blank)**

[4.95](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvb2JqZWN0LW9yaWVudGVkLXByb2dyYW1taW5nLWphdmE/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Enrolled: 5618](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvb2JqZWN0LW9yaWVudGVkLXByb2dyYW1taW5nLWphdmE/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvb2JqZWN0LW9yaWVudGVkLXByb2dyYW1taW5nLWphdmE/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[SRIKANTH VARMA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZGVlcC1sZWFybmluZy1mcmVlLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

**[Deep Learning Course: Deep Dive into Deep Learning](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZGVlcC1sZWFybmluZy1mcmVlLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZGVlcC1sZWFybmluZy1mcmVlLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 2951](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZGVlcC1sZWFybmluZy1mcmVlLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZGVlcC1sZWFybmluZy1mcmVlLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[SRIKANTH VARMA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1jb21wdXRlci1uZXR3b3Jrcy1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

**[Computer Networking Course: Master Computer Networking](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1jb21wdXRlci1uZXR3b3Jrcy1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1jb21wdXRlci1uZXR3b3Jrcy1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Enrolled: 7618](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1jb21wdXRlci1uZXR3b3Jrcy1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1jb21wdXRlci1uZXR3b3Jrcy1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[SRIKANTH VARMA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1zdXBlcnZpc2VkLWxlYXJuaW5nLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

**[Supervised Machine Learning Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1zdXBlcnZpc2VkLWxlYXJuaW5nLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1zdXBlcnZpc2VkLWxlYXJuaW5nLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 5369](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1zdXBlcnZpc2VkLWxlYXJuaW5nLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1zdXBlcnZpc2VkLWxlYXJuaW5nLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[SRIKANTH VARMA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWF0aGVtYXRpY3MtZm9yLW1hY2hpbmUtbGVhcm5pbmctZnJlZS1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

**[Free Maths for Machine Learning Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWF0aGVtYXRpY3MtZm9yLW1hY2hpbmUtbGVhcm5pbmctZnJlZS1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWF0aGVtYXRpY3MtZm9yLW1hY2hpbmUtbGVhcm5pbmctZnJlZS1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Enrolled: 3871](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWF0aGVtYXRpY3MtZm9yLW1hY2hpbmUtbGVhcm5pbmctZnJlZS1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWF0aGVtYXRpY3MtZm9yLW1hY2hpbmUtbGVhcm5pbmctZnJlZS1jb3Vyc2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[JAMSHAID SOHAIL](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0b3JjaC1mb3ItZGVlcC1sZWFybmluZy1mcmVlLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

**[PyTorch for Deep Learning Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0b3JjaC1mb3ItZGVlcC1sZWFybmluZy1mcmVlLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)**

[4.8](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0b3JjaC1mb3ItZGVlcC1sZWFybmluZy1mcmVlLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 1692](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0b3JjaC1mb3ItZGVlcC1sZWFybmluZy1mcmVlLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0b3JjaC1mb3ItZGVlcC1sZWFybmluZy1mcmVlLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[SRIKANTH VARMA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1vcGVyYXRpbmctc3lzdGVtLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

**[Operating System Course: Learn Fundamentals of Operating System](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1vcGVyYXRpbmctc3lzdGVtLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1vcGVyYXRpbmctc3lzdGVtLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 8276](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1vcGVyYXRpbmctc3lzdGVtLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS1vcGVyYXRpbmctc3lzdGVtLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[SRIKANTH VARMA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS11bnN1cGVydmlzZWQtbGVhcm5pbmctY291cnNlP3V0bV9zb3VyY2U9aWI=" \t "_blank)

**[Unsupervised Machine Learning Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS11bnN1cGVydmlzZWQtbGVhcm5pbmctY291cnNlP3V0bV9zb3VyY2U9aWI=" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS11bnN1cGVydmlzZWQtbGVhcm5pbmctY291cnNlP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Enrolled: 1906](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS11bnN1cGVydmlzZWQtbGVhcm5pbmctY291cnNlP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZnJlZS11bnN1cGVydmlzZWQtbGVhcm5pbmctY291cnNlP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[YASH SINHA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLWZvci1kYXRhLXNjaWVuY2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

**[Data Science Course - Mastering the Fundamentals](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLWZvci1kYXRhLXNjaWVuY2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)**

[4.7](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLWZvci1kYXRhLXNjaWVuY2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Enrolled: 13891](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLWZvci1kYXRhLXNjaWVuY2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLWZvci1kYXRhLXNjaWVuY2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[PRATEEK NARANG](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWF0aHMtZm9yLXByb2dyYW1tZXJzP3V0bV9zb3VyY2U9aWI=" \t "_blank)

**[Maths for Programmers](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWF0aHMtZm9yLXByb2dyYW1tZXJzP3V0bV9zb3VyY2U9aWI=" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWF0aHMtZm9yLXByb2dyYW1tZXJzP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Enrolled: 4840](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWF0aHMtZm9yLXByb2dyYW1tZXJzP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWF0aHMtZm9yLXByb2dyYW1tZXJzP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[ARNAV GUPTA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvamF2YS1zcHJpbmctYm9vdC1ibG9nZ2luZy1hcHA/dXRtX3NvdXJjZT1pYg==" \t "_blank)

**[Spring Boot Course: Certified Course for Essential Skills](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvamF2YS1zcHJpbmctYm9vdC1ibG9nZ2luZy1hcHA/dXRtX3NvdXJjZT1pYg==" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvamF2YS1zcHJpbmctYm9vdC1ibG9nZ2luZy1hcHA/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Enrolled: 8998](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvamF2YS1zcHJpbmctYm9vdC1ibG9nZ2luZy1hcHA/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvamF2YS1zcHJpbmctYm9vdC1ibG9nZ2luZy1hcHA/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[GAURAV SISODIA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uva2VyYXMtdGVuc29yZmxvdy1mb3ItZGVlcC1sZWFybmluZz91dG1fc291cmNlPWli" \t "_blank)

**[Keras & TensorFlow for Deep Learning](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uva2VyYXMtdGVuc29yZmxvdy1mb3ItZGVlcC1sZWFybmluZz91dG1fc291cmNlPWli" \t "_blank)**

[4.8](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uva2VyYXMtdGVuc29yZmxvdy1mb3ItZGVlcC1sZWFybmluZz91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 1913](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uva2VyYXMtdGVuc29yZmxvdy1mb3ItZGVlcC1sZWFybmluZz91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uva2VyYXMtdGVuc29yZmxvdy1mb3ItZGVlcC1sZWFybmluZz91dG1fc291cmNlPWli" \t "_blank)

[PRATEEK NARANG](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvY29kaW5nLWxvZ2ljLWJ1aWxkaW5nLWJlZ2lubmVycz91dG1fc291cmNlPWli" \t "_blank)

**[Coding Essentials: Learn Logic Building for Beginners Free Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvY29kaW5nLWxvZ2ljLWJ1aWxkaW5nLWJlZ2lubmVycz91dG1fc291cmNlPWli" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvY29kaW5nLWxvZ2ljLWJ1aWxkaW5nLWJlZ2lubmVycz91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 7002](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvY29kaW5nLWxvZ2ljLWJ1aWxkaW5nLWJlZ2lubmVycz91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvY29kaW5nLWxvZ2ljLWJ1aWxkaW5nLWJlZ2lubmVycz91dG1fc291cmNlPWli" \t "_blank)

[ARNAV GUPTA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWFjaGluZS1jb2RpbmctdGljLXRhYy10b2UtbG93LWxldmVsLWRlc2lnbj91dG1fc291cmNlPWli" \t "_blank)

**[Machine Coding Tic Tac Toe - LLD Case Study](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWFjaGluZS1jb2RpbmctdGljLXRhYy10b2UtbG93LWxldmVsLWRlc2lnbj91dG1fc291cmNlPWli" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWFjaGluZS1jb2RpbmctdGljLXRhYy10b2UtbG93LWxldmVsLWRlc2lnbj91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 1626](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWFjaGluZS1jb2RpbmctdGljLXRhYy10b2UtbG93LWxldmVsLWRlc2lnbj91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvbWFjaGluZS1jb2RpbmctdGljLXRhYy10b2UtbG93LWxldmVsLWRlc2lnbj91dG1fc291cmNlPWli" \t "_blank)

[MRINAL BHATTACHARYA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvbm9kZWpzP3V0bV9zb3VyY2U9aWI=" \t "_blank)

**[Node JS Certification Course - Master the Fundamentals](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvbm9kZWpzP3V0bV9zb3VyY2U9aWI=" \t "_blank)**

[4.8](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvbm9kZWpzP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Enrolled: 13630](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvbm9kZWpzP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvbm9kZWpzP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[PRATEEK NARANG](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3FsLXVzaW5nLW15c3FsLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

**[SQL for Beginners: Learn SQL using MySQL and Database Design Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3FsLXVzaW5nLW15c3FsLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3FsLXVzaW5nLW15c3FsLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 19563](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3FsLXVzaW5nLW15c3FsLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3FsLXVzaW5nLW15c3FsLWNvdXJzZT91dG1fc291cmNlPWli" \t "_blank)

[ADITYA JAIN](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvY3BwLWRhdGEtc3RydWN0dXJlcz91dG1fc291cmNlPWli" \t "_blank)

**[Data Structures in C++ Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvY3BwLWRhdGEtc3RydWN0dXJlcz91dG1fc291cmNlPWli" \t "_blank)**

[4.5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvY3BwLWRhdGEtc3RydWN0dXJlcz91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 16423](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvY3BwLWRhdGEtc3RydWN0dXJlcz91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvY3BwLWRhdGEtc3RydWN0dXJlcz91dG1fc291cmNlPWli" \t "_blank)

[SUBHESH KUMAR](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHNhLWJlZ2lubmVycy1qYXZhP3V0bV9zb3VyY2U9aWI=" \t "_blank)

**[Java DSA Course - Master the Fundamentals and Beyond](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHNhLWJlZ2lubmVycy1qYXZhP3V0bV9zb3VyY2U9aWI=" \t "_blank)**

[4.95](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHNhLWJlZ2lubmVycy1qYXZhP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Enrolled: 25623](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHNhLWJlZ2lubmVycy1qYXZhP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHNhLWJlZ2lubmVycy1qYXZhP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[JITENDER PUNIA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHNhLWludGVydmlld3MtamF2YT91dG1fc291cmNlPWli" \t "_blank)

**[DSA Problem Solving for Interviews using Java](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHNhLWludGVydmlld3MtamF2YT91dG1fc291cmNlPWli" \t "_blank)**

[4.9](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHNhLWludGVydmlld3MtamF2YT91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 40457](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHNhLWludGVydmlld3MtamF2YT91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHNhLWludGVydmlld3MtamF2YT91dG1fc291cmNlPWli" \t "_blank)

[SRIKANTH VARMA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZGJtcz91dG1fc291cmNlPWli" \t "_blank)

**[DBMS Course - Master the Fundamentals and Advanced Concepts](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZGJtcz91dG1fc291cmNlPWli" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZGJtcz91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 47529](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZGJtcz91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZGJtcz91dG1fc291cmNlPWli" \t "_blank)

[SRIKANTH VARMA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLXNxbC1kYXRhLXNjaWVuY2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

**[Python and SQL for Data Science](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLXNxbC1kYXRhLXNjaWVuY2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLXNxbC1kYXRhLXNjaWVuY2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Enrolled: 44366](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLXNxbC1kYXRhLXNjaWVuY2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcHl0aG9uLXNxbC1kYXRhLXNjaWVuY2U/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[SRIKANTH VARMA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZWRhLWFuZC1kYXRhLXZpc3VhbGlzYXRpb24/dXRtX3NvdXJjZT1pYg==" \t "_blank)

**[EDA and Data Visualization Course in Data Science](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZWRhLWFuZC1kYXRhLXZpc3VhbGlzYXRpb24/dXRtX3NvdXJjZT1pYg==" \t "_blank)**

[5](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZWRhLWFuZC1kYXRhLXZpc3VhbGlzYXRpb24/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Enrolled: 7254](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZWRhLWFuZC1kYXRhLXZpc3VhbGlzYXRpb24/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZWRhLWFuZC1kYXRhLXZpc3VhbGlzYXRpb24/dXRtX3NvdXJjZT1pYg==" \t "_blank)

[TARUN MALHOTRA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHluYW1pYy1wcm9ncmFtbWluZz91dG1fc291cmNlPWli" \t "_blank)

**[Dynamic Programming Course - Learn Optimizing Complex Problems](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHluYW1pYy1wcm9ncmFtbWluZz91dG1fc291cmNlPWli" \t "_blank)**

[4.9](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHluYW1pYy1wcm9ncmFtbWluZz91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 6267](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHluYW1pYy1wcm9ncmFtbWluZz91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvZHluYW1pYy1wcm9ncmFtbWluZz91dG1fc291cmNlPWli" \t "_blank)

[SATYA SAI](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3RyaW5nLXBhdHRlcm4tbWF0Y2hpbmcta21wLWFsZ29yaXRobT91dG1fc291cmNlPWli" \t "_blank)

**[String Pattern Matching: KMP Algorithm](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3RyaW5nLXBhdHRlcm4tbWF0Y2hpbmcta21wLWFsZ29yaXRobT91dG1fc291cmNlPWli" \t "_blank)**

[4.95](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3RyaW5nLXBhdHRlcm4tbWF0Y2hpbmcta21wLWFsZ29yaXRobT91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 3854](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3RyaW5nLXBhdHRlcm4tbWF0Y2hpbmcta21wLWFsZ29yaXRobT91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3RyaW5nLXBhdHRlcm4tbWF0Y2hpbmcta21wLWFsZ29yaXRobT91dG1fc291cmNlPWli" \t "_blank)

[MOHIT YADAV](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3FsLXZzLW5vc3FsP3V0bV9zb3VyY2U9aWI=" \t "_blank)

**[SQL v/s NoSQL Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3FsLXZzLW5vc3FsP3V0bV9zb3VyY2U9aWI=" \t "_blank)**

[4.8](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3FsLXZzLW5vc3FsP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Enrolled: 8445](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3FsLXZzLW5vc3FsP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2Uvc3FsLXZzLW5vc3FsP3V0bV9zb3VyY2U9aWI=" \t "_blank)

[TARUN MALHOTRA](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcXVvcmEtc3lzdGVtLWRlc2lnbj91dG1fc291cmNlPWli" \t "_blank)

**[Quora System Design Course](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcXVvcmEtc3lzdGVtLWRlc2lnbj91dG1fc291cmNlPWli" \t "_blank)**

[4.9](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcXVvcmEtc3lzdGVtLWRlc2lnbj91dG1fc291cmNlPWli" \t "_blank)

[Enrolled: 2522](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcXVvcmEtc3lzdGVtLWRlc2lnbj91dG1fc291cmNlPWli" \t "_blank)

[Free](https://www.interviewbit.com/api/v3/redirect/scaler_auth/?redirect_url=aHR0cHM6Ly9zY2FsZXIuY29tL3RvcGljcy9jb3Vyc2UvcXVvcmEtc3lzdGVtLWRlc2lnbj91dG1fc291cmNlPWli" \t "_blank)

**6. Write a program in Java to Toggle the case of every character of a string. For instance, if the input string is “ApPLe”, the output should be “aPplE”.**

We know that in Java, we cannot make changes to the same string as it is immutable. So, we have to return a new String. The lowercase ASCII characters differ from the uppercase ASCII characters by 32. This means ‘a’ - 32 = ‘A’. So, we will use this concept to Toggle the String cases.

**Java Code to Toggle Cases**

**import** java.util.\*;

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

String str = scn.nextLine();

StringBuilder res = **new** StringBuilder("");

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

**char** ch = str.charAt(i); //current character

**if**(ch >='A' && ch <= 'Z') {

res.append((**char**)(ch + 32));

} **else** **if**(ch >='a' && ch<='z'){

res.append((**char**)(ch - 32));

} **else** {

res.append(ch);

}

}

String ans = res.toString();

System.out.println("The string after toggling becomes: " + ans);

}

}

**Sample Output:**

Input: Ab#$Cd

Output: aB#$cD

* **Corner Cases, You Might Miss:**The String can contain other characters apart from the alphabet. So, in that case, we do not have to change those characters that are not alphabets, while we have to toggle the alphabets. Hence, in the code, after the if condition, we have an else-if condition and not the else condition; otherwise it would have subtracted 32 from every character that is not an uppercase alphabet. In the else condition, we have added the character as it is. This is also seen in the output shown above.
* **Time Complexity:**Since we have used a StringBuilder in place of a String, the time complexity of inserting a character in a StringBuilder is O(1). Since we are inserting N characters, the time complexity is O(N). {Here N is the length of the input string}
* **Auxiliary Space:**O(1) as we have not used any extra space to solve the problem. The string ans and StringBuilder res are the output spaces and not the auxiliary space.

**7. Write a program in Java to count the total number of vowels and consonants in a String. The string can contain all the alphanumeric and other special characters as well. However, only the lowercase English alphabets are allowed in the String.**

We just have to traverse the string. If we get any vowel (a,e,i,o,u), we increment the variable corresponding to the vowel count and if we get a consonant, we increment the variable corresponding to the consonant count.

**Java Code to Count Vowels and Consonants in a String**

**import** java.util.\*;

**class** **Main** {

**public** **static** **boolean** **isVowel**(**char** ch) {

**if**(ch == 'a' || ch =='e' || ch =='i' || ch =='o' || ch =='u')

**return** **true**;

**return** **false**;

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

String str = scn.nextLine();

**int** vowelCount = 0;

**int** consCount = 0;

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

**char** ch = str.charAt(i);

**if**(isVowel(ch) == **true**) vowelCount++;

**else** **if**(ch >='a' && ch<='z' && isVowel(ch) == **false**) consCount++;

}

System.out.println("Number of vowels are: " + vowelCount);

System.out.println("Number of consonants are: " + consCount);

System.out.println("Number of other characters are: " + (**int**)(str.length() - vowelCount -consCount));

}

}

**Sample Output**

Input: ae#zyu\*

Output:

The number of vowels is: 3

The number of consonants is: 2

The number of other characters is: 2

* **Corner Cases, You Might Miss:**In order to check whether a character is a vowel or not, we have a function. However, it is not right to say that if it is not a vowel then it will be a consonant as it can also be any other character. So, we have to make sure that it is an alphabet and then make sure that it is not a vowel. The same is done in the code.
* **Time Complexity:**O(N) where N is the length of the input string as we have to traverse the entire string once.
* **Auxiliary Space:**O(1) as we have not used any extra space.

Advance your career with  **Mock Assessments**

Real-world coding challenges for top company interviews

Real-Life Problems

Detailed reports

**Attempt Now**

**8. Write a program to print all the unique characters in a String. For instance, if the input string is “abcb”, the output will be the characters ‘a’ and ‘c’ as they are unique. The character ‘b’ repeats twice and so it will not be printed.**

We can use a HashSet in order to store the characters of the String. When we arrive at a character in the String, if it is already present in the HashSet, we remove it from the HashSet as that character is not unique. If it is not present inside the HashSet, we add it to it. After traversing the entire string, we print the elements inside the HashMap.

**Java Code to Print All Unique Characters in a String.**

**import** java.util.\*;

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

String str = scn.nextLine();

HashSet<Character> unique = **new** HashSet<>();

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

**char** ch = str.charAt(i);

**if**(unique.contains(ch) == **true**) {

//this character has already occured

unique.remove(ch);

} **else** {

unique.add(ch);

}

}

**if**(unique.size() == 0) {

System.out.println("There are no unique characters");

}

**for**(Character ch : unique) {

System.out.print(ch + " ");

}

}

}

**Sample Output**

Input: abcab

Output: c

* **Corner Cases, You Might Miss:**What if such a string is passed that has all the duplicate characters? Also, it might happen that an empty string is passed as the input. So, in such a case, the size of the HashSet will remain 0 after processing the String. Hence, we have handled the case of a hashset having 0 sizes separately.
* **Time Complexity:**The time complexity is O(N) where N is the length of the string as we are traversing the entire string.
* **Auxiliary Space:**O(N) as it might happen that all the N characters are unique so, we might generate a HashSet of size N.

**9. Write a program in Java to prove that the strings are immutable in Java.**

Strings are immutable in Java. This can be proven by making changes to a string and comparing them with the == operator. Since this operator compares only the references, i.e. the addresses of the objects, it will be able to tell us if the changes are made to the same object or not. If after making changes to a string we compare it by == and we get no equals, this means that the strings are immutable.

**Java program to prove Strings are Immutable**

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

String s1 = "InterviewBit";

String s2 = s1;

System.out.println(s1 == s2); //they are equal

s1 += "Scaler";

System.out.println(s1 == s2); //not equal

}

}

**Output**

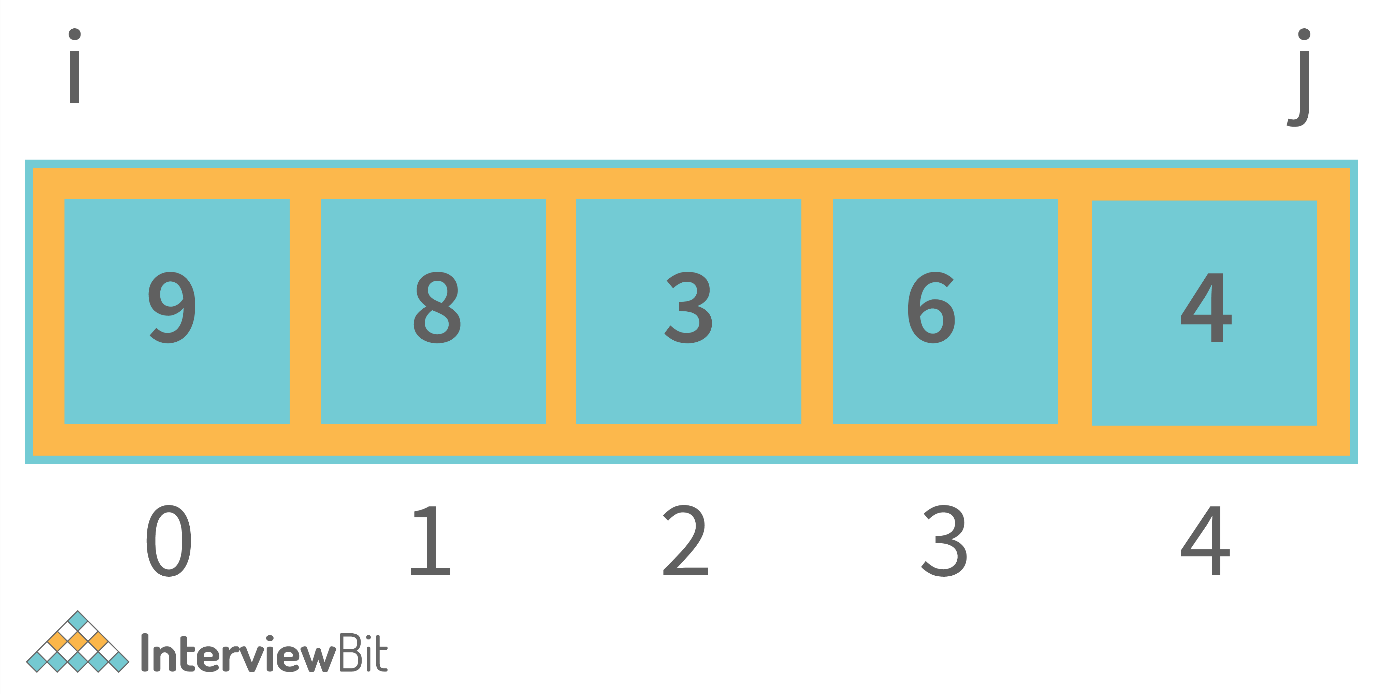
**true**

**false**

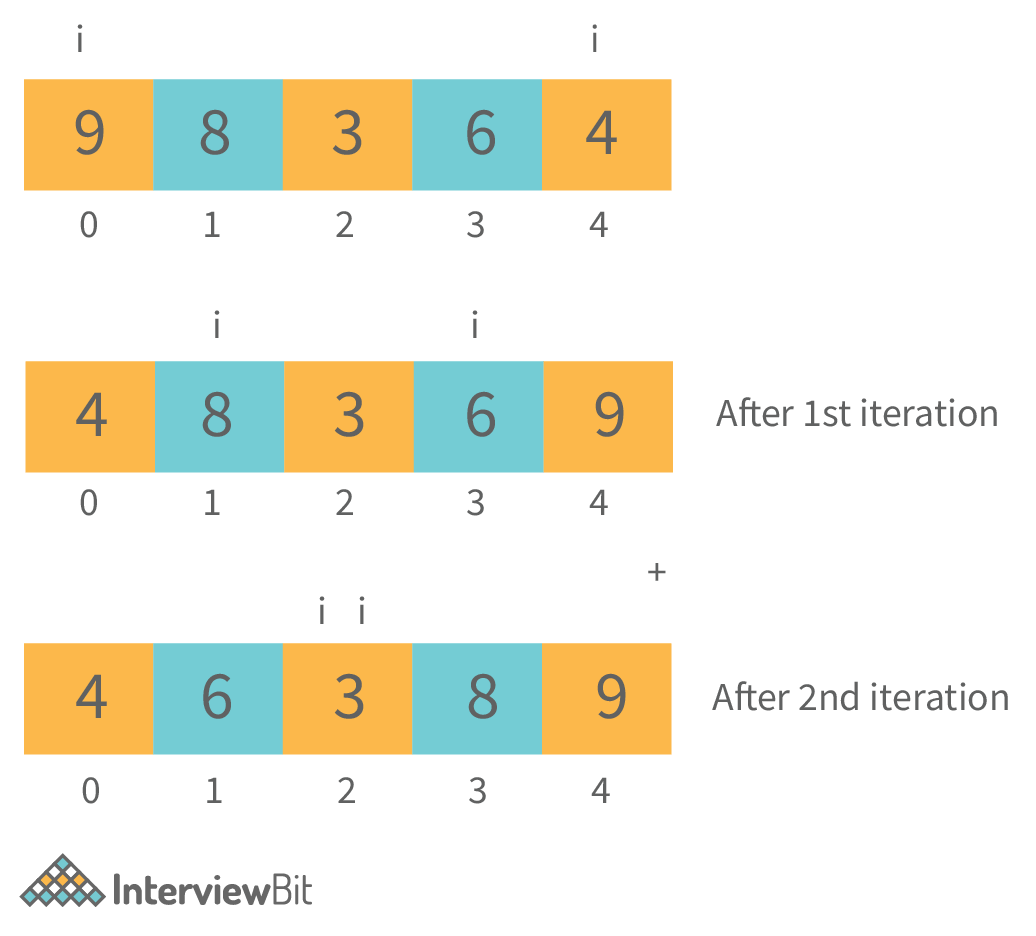
So, let us now discuss an array of programs related to Java interviews.

**10. Write a program in Java to Reverse an Array without using extra space.**

We keep to variables i and j at both the ends of the array as shown below.



Now, swap the element at index i with the element at index j increment the i variable, and decrement the j variable. Keep doing this till i is less than j. This is shown in the image below.



**import** java.util.\*;

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**int** n = scn.nextInt();

**int**[] arr = **new** **int**[n];

**for**(**int** i=0;i<n;i++) {

arr[i] = scn.nextInt();

}

System.out.println("The reversed array is");

**int** i = 0;

**int** j = arr.length - 1;

**while**(i < j) {

//swapping ith and jth index elements

**int** temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

i++;

j--;

}

//displaying the array

**for**(**int** it=0;it<arr.length;it++) {

System.out.print(arr[it] + " ");

}

}

}

**Sample Output**

Input:

5

1 2 3 4 5

Output:

The reversed array is

5 4 3 2 1

* **Time complexity:**O(N) as we are traversing the array.
* **Auxiliary Space:**O(1) as we have not used any extra space.

**11. Write a program to find the index of first occurrence and last occurrence of an element in the array in a single iteration.**

**You cannot use extra space. The first index of occurrence and the last index of occurrence will be -1, if the element is not present inside the array.**

We will keep three variables. The 2 variables (that are) firstIndex and lastIndex will be initialized to -1. The third will be a boolean type variable found which will be initially false. If the element is found, we will make it true, and store the current index in firstIndex and lastIndex variables. If the element is found further, only the lastIndex variable will change. The fact that the number has already been found in the array will be denoted by the found boolean variable.

**Java Program to find the First and Last Occurrence of an element in the Array**

**import** java.util.\*;

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**int** n = scn.nextInt();

**int**[] arr = **new** **int**[n];

// input the array

**for**(**int** i=0;i<n;i++) {

arr[i] = scn.nextInt();

}

**int** target = scn.nextInt();

**int** fIndex = -1, lIndex = -1;

**boolean** found = **false**;

**for**(**int** i=0;i<n;i++) {

**if**(arr[i] == target) {

**if**(!found) {

fIndex = i;

lIndex = i;

found = **true**; //found for the first time

} **else** {

lIndex = i;

}

}

}

**if**(found == **false**) {

System.out.println("The element does not exist in the array");

} **else** {

System.out.println("First Index = " + fIndex + " Last Index = " + lIndex);

}

}

}

**Sample Output**

When element exists

Input:

5

1 2 3 2 5

2

Output: First Index = 1 Last Index = 3

When the element does not exist

Input:

5

1 2 3 2 5

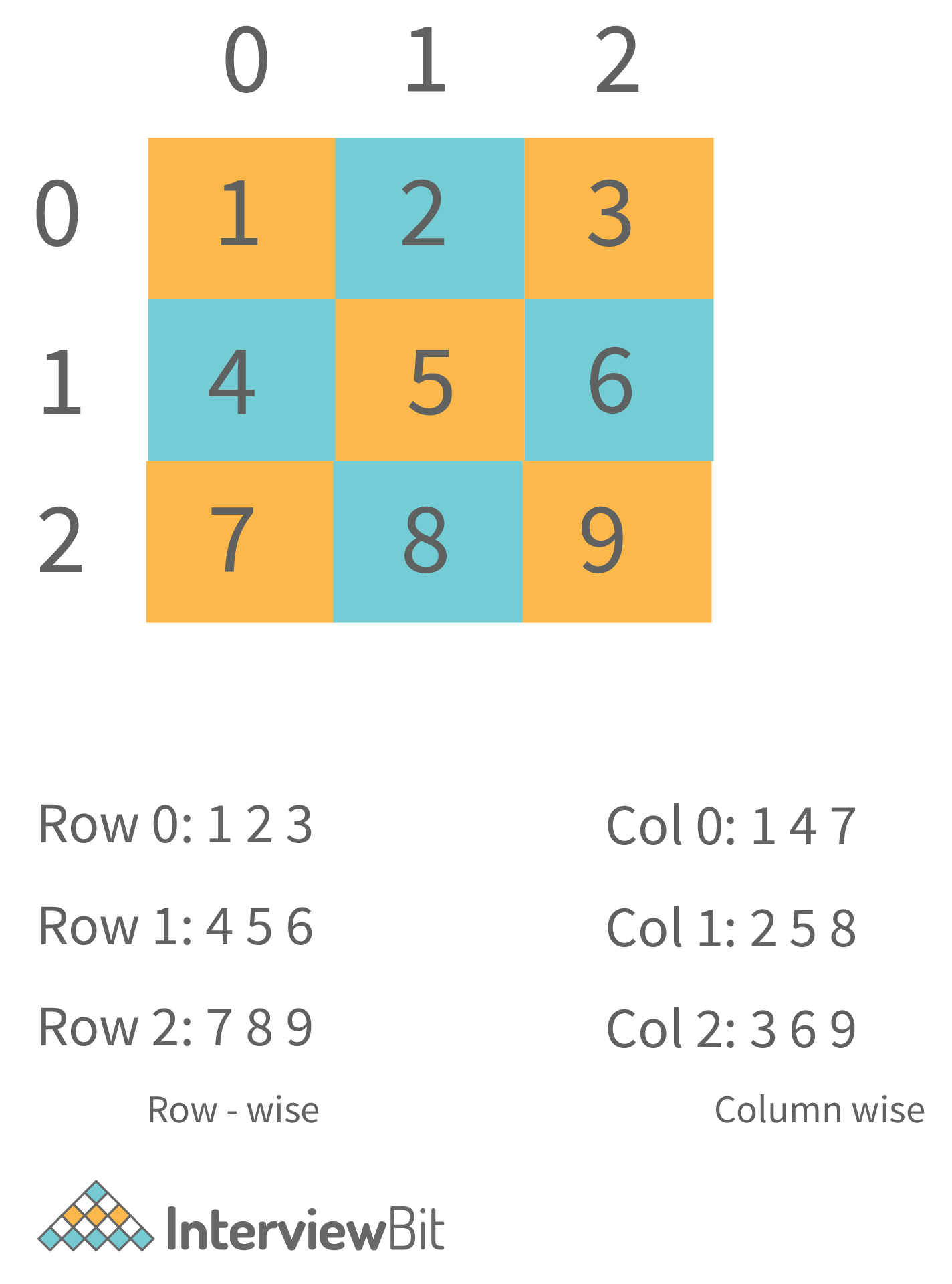
2

Output:The element does not exist in the array.

* **Corner Cases, You might Miss:**The corner case of elements not present in the array is something that should be taken care of separately. In our code, we use the boolean variable found to do so. Otherwise, we can directly see if the values of fIndex and lIndex variables are -1 or not.
* **Time Complexity:**O(N) as we are traversing the array.
* **Auxiliary Space:**O(1) as we have not used any extra space to solve the problem.

**12. Write a program in Java to input an NxN matrix and display it row-wise and column-wise.**

Simply input the matrix. Now, display the matrix row-wise by starting from the first row and moving to the next elements within the same row. For displaying column-wise, start from the first column and keep moving in the same column to the next elements. This is shown below.



**Java Code to input and display 2-D Matrix**

**import** java.util.\*;

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**int** N = scn.nextInt();

**int**[][] mat = **new** **int**[N][N];

**for**(**int** i=0;i<N;i++) {

**for**(**int** j=0;j<N;j++) {

mat[i][j] = scn.nextInt();

}

}

//Display Row wise

**for**(**int** i=0;i<N;i++) {

System.out.print("Row " + i + " : ");

**for**(**int** j=0;j<N;j++) {

System.out.print(mat[i][j] + " ");

}

System.out.println("\t");

}

System.out.println();

//Display Col wise

**for**(**int** j=0;j<N;j++) {

System.out.print("Col " + j + " : ");

**for**(**int** i=0;i<N;i++) {

System.out.print(mat[i][j] + " ");

}

System.out.println("\t");

}

}

}

**Sample Output**

Input:

3

1 2 3

4 5 6

7 8 9

Output:

Row 0 : 1 2 3

Row 1 : 4 5 6

Row 2 : 7 8 9

Col 0 : 1 4 7

Col 1 : 2 5 8

Col 2 : 3 6 9

* **Time Complexity:**O(N to the power 2) as we traverse the 2-D array to print it.
* **Auxiliary Space:**O(1) as we have not used any extra space.

Get Access to 250+ Guides with Scaler Mobile App!

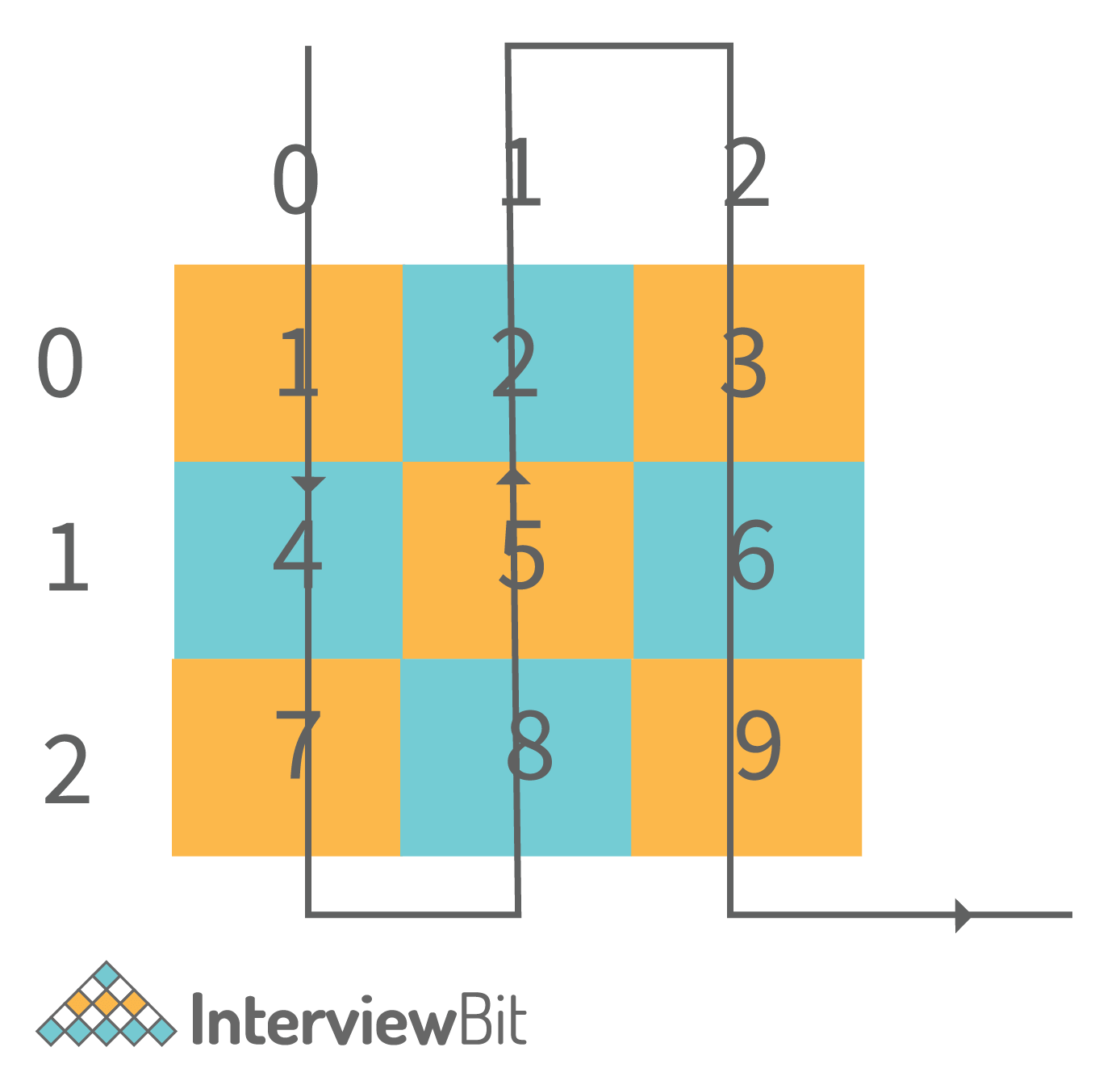
Experience free learning content on the Scaler Mobile App

4.5

100K+

Play Store

**13. Write a program in Java to print the elements of the matrix in Wave Order as shown below. (The matrix can have different numbers of rows and columns).**



It is clear from the image itself that we are traversing column-wise. Now, when we traverse an even column, we traverse it from top to bottom and when we traverse an odd column, we traverse it from bottom to top direction.

**Code for Wave Print a Matrix in Java**

**import** java.io.\*;

**import** java.util.\*;

**public** **class** **Main**{

**public** **static** **void** **main**(String[] args) **throws** Exception {

// write your code here

Scanner scn = **new** Scanner(System.in);

**int** n = scn.nextInt();

**int** m = scn.nextInt();

**int**[][] mat = **new** **int**[n][m];

//input the matrix

**for**(**int** i=0;i<n;i++) {

**for**(**int** j=0;j<m;j++) {

mat[i][j] = scn.nextInt();

}

}

**for**(**int** j=0;j<mat[0].length;j++) {

**if**(j% 2 == 0) {

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

System.out.print(mat[i][j] + " ");

}

} **else** {

**for**(**int** i=mat.length-1;i>=0;i--) {

System.out.print(mat[i][j] + " ");

}

}

System.out.println();

}

}

}

**Sample Output**

Input:

1 2 3

4 5 6

7 8 9

Output:

1 4 7

8 5 2

3 6 9

* **Time Complexity:**O(N \* M) where N is the number of rows and M is the number of columns.
* **Auxiliary Space:**O(1) as we have not used any extra space to solve this problem.

**14. Write a class “Programmer”. Give some properties and methods to it and show how you will access them in the main method by creating object(s) of this class.**

The following is the example code.

**Java Code for Custom Class**

**import** java.util.\*;

**class** **Programmer** {

**private** **int** age;

**private** String name;

Programmer() {

}

Programmer(**int** age, String name) {

**this**.age = age;

**this**.name = name;

}

**void** **setAge**(**int** age) {

**this**.age = age;

}

**void** **setName**(String name) {

**this**.name = name;

}

**int** **getAge**() {

**return** age;

}

String **getName**() {

**return** name;

}

**public** **void** **codes**() {

System.out.println(**this**.name + " writes codes");

}

**public** **void** **drinksCoffee**() {

System.out.println(**this**.name + " drinks coffee and can then convert exponential complexity codes to polynomial");

}

}

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Programmer p1 = **new** Programmer(22,"Guneet");

p1.codes();

p1.drinksCoffee();

}

}

**Sample Output**

Guneet writes codes

Guneet drinks coffee and can then convert exponential complexity codes to polynomial.

* **Some things you should keep in mind:**The properties must usually be set private and we should have getter and setter methods to access and modify them. This is good OOPS practice. Also, always create a default constructor as when we create a parameterized constructor, Java removes its own default constructor and the object creation without passing the parameters to the constructor would not be possible.

**15. Write a program in Java to show inheritance in Java.**

This program is for testing the concepts of inheritance in Java and the usage of extends keyword. Following is an example program in which a class SmartPhone extends a class Phone. This is a real-life example as a phone has basic features of calling and messaging whereas a smartphone has several other features like clicking pictures, playing music, etc.

**Java Code for showing Inheritance**

class Phone {

private int number;

Phone() {

}

void setNumber(int number) {

this.number = number;

}

int getNumber() {

return number;

}

public void call() {

System.out.println("Call is made");

}

public void message() {

System.out.println("Message is sent");

}

}

class SmartPhone extends Phone {

int cameraMegaPX;

public void click() {

System.out.println("A photograph was clicked");

}

public void playMusic() {

System.out.println("Music Started Playing");

}

public void pauseMusic() {

System.out.println("Music Paused");

}

public void stopMusic() {

System.out.println("Music Stopped");

}

}

class Main {

public static void main(String args[]) {

// Your code goes here

SmartPhone p1 = new SmartPhone();

p1.setNumber(9863472);

System.out.println("Phone number is: " + p1.getNumber());

p1.call();

p1.playMusic();

}

}

**Sample Output**

Phone number is: 9863472

Call is made

Music Started Playing

**16. Write a program in Java to show a basic “divide by 0 exception”.**

Divide by zero exception occurs when we try to divide a number by 0 in Java. Following is the program showing divide by 0 exception.

**import** java.util.\*;

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**int** n = scn.nextInt();

System.out.println("Dividing this number by 0");

**try** {

System.out.println(n/0);

} **catch**(Exception e) {

System.out.println(e);

}

System.out.println("Program completed");

}

}

**Output**

Input: 3

Output:

Dividing **this** number by 0

java.lang.ArithmeticException: / by zero

Program completed

**Important:**In this question, we used the try and catch block for handling the divide by 0 exception. Hence, the complete execution of the program took place. Otherwise, the program would have stopped at the exception, and “Program completed” would not have been printed.

**17. Write a program to show a single thread in Java.**

**Java Program to show Single Thread**

**public** **class** **Main** {

**public** **static** **void** **main**(String[] args) {

Thread t = Thread.currentThread();

t.setName("My Main Thread");

t.setPriority(7);

System.out.println(t);

System.out.println(t.getName());

System.out.println(t.getPriority());

}

}

**Output**

Thread[My Main Thread,7,main]

My Main Thread

7

**Java Programming Interview Questions for Experienced**

**1. A sentence is said to be a palindrome if we convert all its alphabets to lowercase, include the numerics but exclude all the spaces, whitespaces, and other special characters and it reads the same from left to right and right to left.**

**For instance, consider the following sentence: “2 Race, e cAr 2”. This sentence will be converted to “2raceecar2”. The string is a palindrome, hence this sentence is a palindrome. You have to take a sentence input from the user and print “true” if it is a palindrome, or else print “false”.**

 A sentence is said to be a palindrome if we convert all its alphabets to lowercase, include the numerics but exclude all the spaces, whitespaces, and other special characters and it reads the same from left to right and right to left.

So, the approach is pretty simple. We convert the sentence into a string by including all the alphanumeric characters and excluding all the other characters. The alphabets will be included only in their lowercase format. Then, we simply have to check whether a string is a palindrome or not. For this, we keep a pointer “lo” at the beginning of the string and a pointer “hi” at the end of the string. We keep incrementing lo and decrementing hi while checking whether the characters at these indices are equal or not. If at any place, we find that the characters are not equal, the string is not a palindrome. If lo becomes greater than hi and the characters at lo and hi were the same throughout, the string is a palindrome.

**Java Code to check Palindromic Sentence**

**import** java.util.\*;

**class** **Main** {

**public** **static** **boolean** **isStrPalindrome**(String str) {

**int** lo = 0;

**int** hi = str.length()-1;

**while**(lo < hi) {

**char** ch1 = str.charAt(lo);

**char** ch2 = str.charAt(hi);

**if**(ch1 != ch2) **return** **false**;

lo++;

hi--;

}

**return** **true**;

}

**public** **static** **boolean** **isSentencePalindrome**(String s) {

String res = "";

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

**char** ch = s.charAt(i);

**if**((ch >='a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >='0' && ch<='9')) {

**if**(ch >='A' && ch <= 'Z') res += (**char**)(ch + 32);

**else** res += ch;

} **else** **continue**;

}

**if**(isStrPalindrome(res)) **return** **true**;

**return** **false**;

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

String sentence = scn.nextLine();

**if**(isSentencePalindrome(sentence)) System.out.println(**true**);

**else** System.out.println(**false**);

}

}

**Sample Output**

When the sentence is a palindrome

Input: 2 Race, e cAr 2

Output: **true**

When the sentence is not a palindrome

Input: 2 Race, a cAr 2

Output: **false**

* **Corner cases, You Might Miss:**It is very important to convert all the alphabets in the String to lowercase. If this is not done, our answer will not be correct. Also, the special case of the string being empty is not handled separately as the program automatically covers this test case by not including any character of the string. So, according to our program, an empty string will be a palindrome.
* If you want that the answer should be false in the case of an empty string, you can apply this condition in the isStrPalindrome() function.
* **Time Complexity:**O(N) where N is the length of the input string.  
  **Auxiliary Space:**O(1) as we have not used any extra space.

**2. Add two Binary Strings and return a Binary String as a result. The addition should be performed as per the rules of binary addition.**

So, the question is basically to add 2 binary numbers given in the form of strings. We should know the basic rules of binary addition:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 and Carry = 1

This shows that whenever the result exceeds 1, the answer of addition becomes 0 and carry becomes 1. So, using these rules, we will add 2 binary strings starting from their LSBs i.e. from the last index of each string moving towards the first index.

**Java Code for Binary Addition of Strings**

**import** java.util.\*;

**class** **Main** {

**public** **static** String **add**(String a, String b) {

String ans = "";

**if**(a.equals("0") && b.equals("0")) **return** "0";

**int** i = a.length() -1;

**int** j = b.length() -1;

**int** ca = 0;

**while**(i >=0 || j>=0 || ca >0) {

**int** d1 = (i >= 0) ? (a.charAt(i) - '0') : 0;

**int** d2 = (j >= 0) ? (b.charAt(j) - '0') : 0;

**int** digit = 0;

**if**(d1 + d2 + ca >= 2) {

digit = (d1 + d2 + ca) % 2;

ca = (d1 + d2 + ca) / 2;

} **else** {

digit = d1 + d2 + ca;

ca = 0;

}

i--;

j--;

ans = digit + ans;

}

**return** ans;

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

String a = scn.nextLine();

String b = scn.nextLine();

System.out.println("The sum is: " + add(a,b));

}

}

**Sample Output**

Input:

1

0111

Output: The sum is: 1000

* **Corner Cases, You Might Miss:**It is very important to address that the numbers might not be of equal length. As in the example shown above, the first number is 1 and the second is 0111. So, the first number is smaller than the second number. The second number can also be smaller than the first number. Also, even if the numbers of equal lengths are passed, the result of addition can exceed one bit. As in the example shown above, the larger number was a 3-bit number 111 and the output is a 4-bit number 1000.
* **Time Complexity:**O(N) where N is the length of the longer string among the 2 input binary strings.
* **Auxiliary Space:**O(1) as we have not used any extra space to solve our problem.

**3. You are given 2 strings as input. You have to check whether they are anagrams or not.**

**Anagrams are those strings that have the same characters occurring an equal number of times in both the strings. However, the order can be different. For example “anagram” and “nagrama” are Anagrams.**

We will use HashMap to store the frequency of each character of the first string. Then, we will traverse the second string and keep on decrementing the frequency in the HashMap. If for any character in the second string, either the character is not present in the HashMap or its frequency is already 0, we will return false. Else, if we have scanned the entire second String and there are no discrepancies, the two strings will be anagrams.

**Java Code to check Anagrams**

**import** java.util.\*;

**class** **Main** {

**public** **static** **boolean** **isAnagram**(String s1, String s2) {

**if**(s1.length() != s2.length()) **return** **false**;

HashMap<Character,Integer> fmap = **new** HashMap<>();

**for**(**int** i=0;i<s1.length();i++) {

**int** ofreq = fmap.getOrDefault(s1.charAt(i),0);

fmap.put(s1.charAt(i),ofreq+1);

}

**for**(**int** i=0;i<s2.length();i++) {

**if**(!fmap.containsKey(s2.charAt(i)) || fmap.get(s2.charAt(i)) == 0) {

**return** **false**;

} **else** {

**int** ofreq = fmap.get(s2.charAt(i));

fmap.put(s2.charAt(i),ofreq-1);

}

}

**return** **true**;

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

String str1 = scn.nextLine();

String str2 = scn.nextLine();

**if**(isAnagram(str1,str2)) System.out.println(**true**);

**else** System.out.println(**false**);

}

}

**Sample Output**

When the strings are anagrams

Input:

anagram

nagrama

Output:  **true**

Input:

anagram

nagrame

Output: **false**

* **Corner Cases, You Might Miss:**Is there any need to check the strings if the length of the strings is not equal? The answer is NO as they don’t have an equal number of characters so, they can never be anagrams. So, a separate check for the length of the strings will be beneficial.
* **Time Complexity:**O(N + M) where N and M are the lengths of the two strings. This is because we have traversed both the strings separately.
* **Auxiliary Space:**O(N) where N is the length of the first string. This is because it might happen that all the N characters in the first String are unique.

**4. You are given a sorted array of integers. It is given that each element in the array is unique.**

**You have to find the index where the element is located in the array. If it is not located in the array, you have to return the index at which it should be inserted in the array so that the array remains sorted. You can’t use extra space and the expected time complexity is O(log2N) where N is the number of elements in the array.**

Since the array is sorted, we will use Binary Search to find the element. If the element is not found, the index at which we insert an element is always the ceil Index. So, what is the ceil index? At the end of the binary search, ceil index is where the low (or left) pointer points. So, the code for the same is shown below.

**Java Code to Search Element/Insert Position**

**import** java.util.\*;

**class** **Main** {

**public** **static** **int** **ceilIndex**(**int**[] nums, **int** target) {

**int** lo = 0;

**int** hi = nums.length-1;

**while**(lo <= hi) {

**int** mid = lo + (hi-lo)/2;

**if**(nums[mid] == target) {

**return** mid;

} **else** **if**(nums[mid] < target) {

lo = mid + 1;

} **else** {

hi = mid - 1;

}

}

**return** lo; //ceil

}

**public** **static** **int** **search**(**int**[] nums, **int** target) {

//insert position is actually the ceil of the element

**return** ceilIndex(nums,target);

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**int** n = scn.nextInt();

**int**[] arr = **new** **int**[n];

**for**(**int** i=0;i<n;i++) {

arr[i] = scn.nextInt();

}

**int** target = scn.nextInt();

System.out.println(search(arr,target));

}

}

**Sample Output**

When the element is present in the array

Input:

4

1 3 5 6

5

Output: 2

When the element is not present in the array

Input:

4

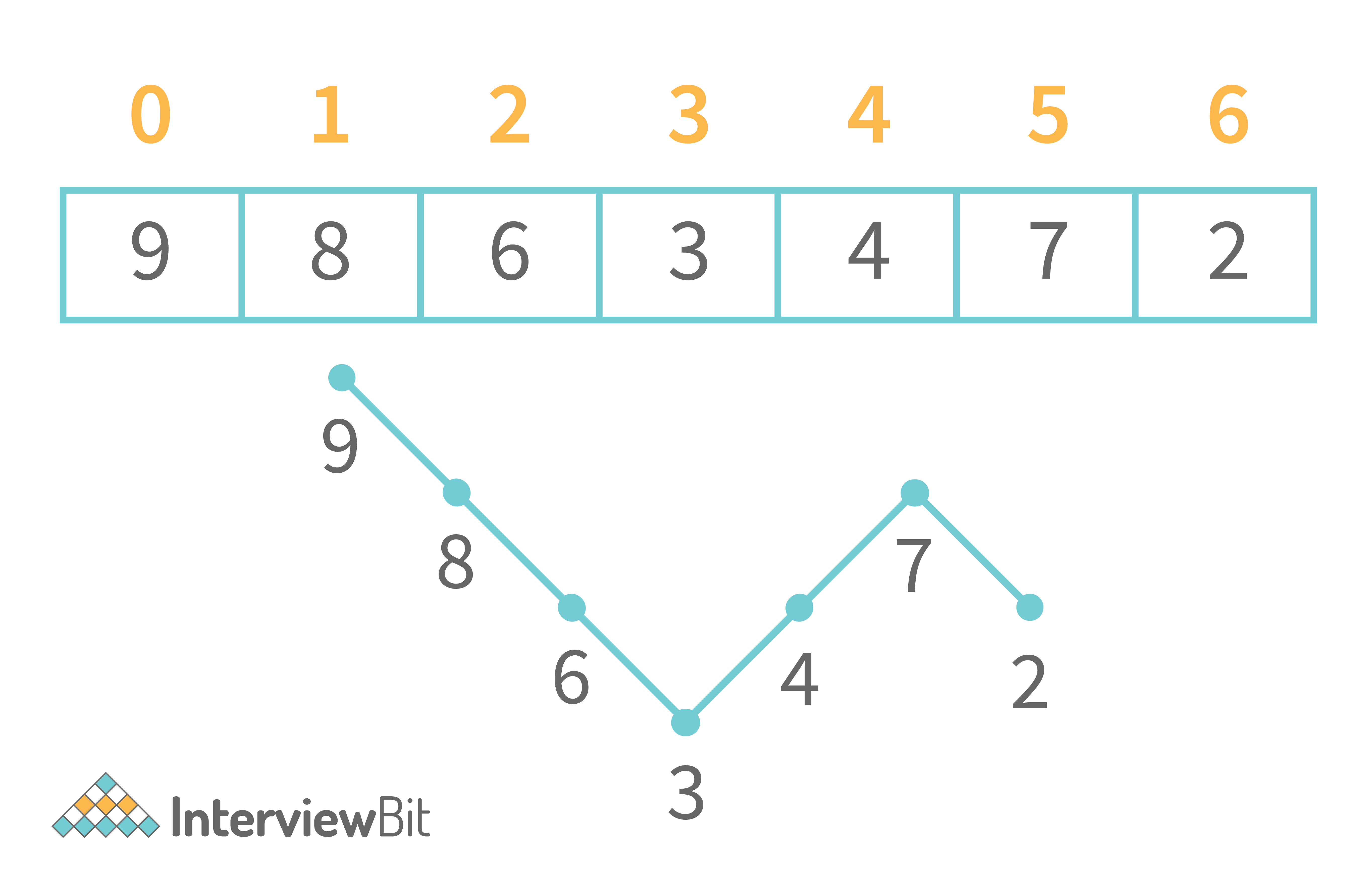
1 3 5 6

4

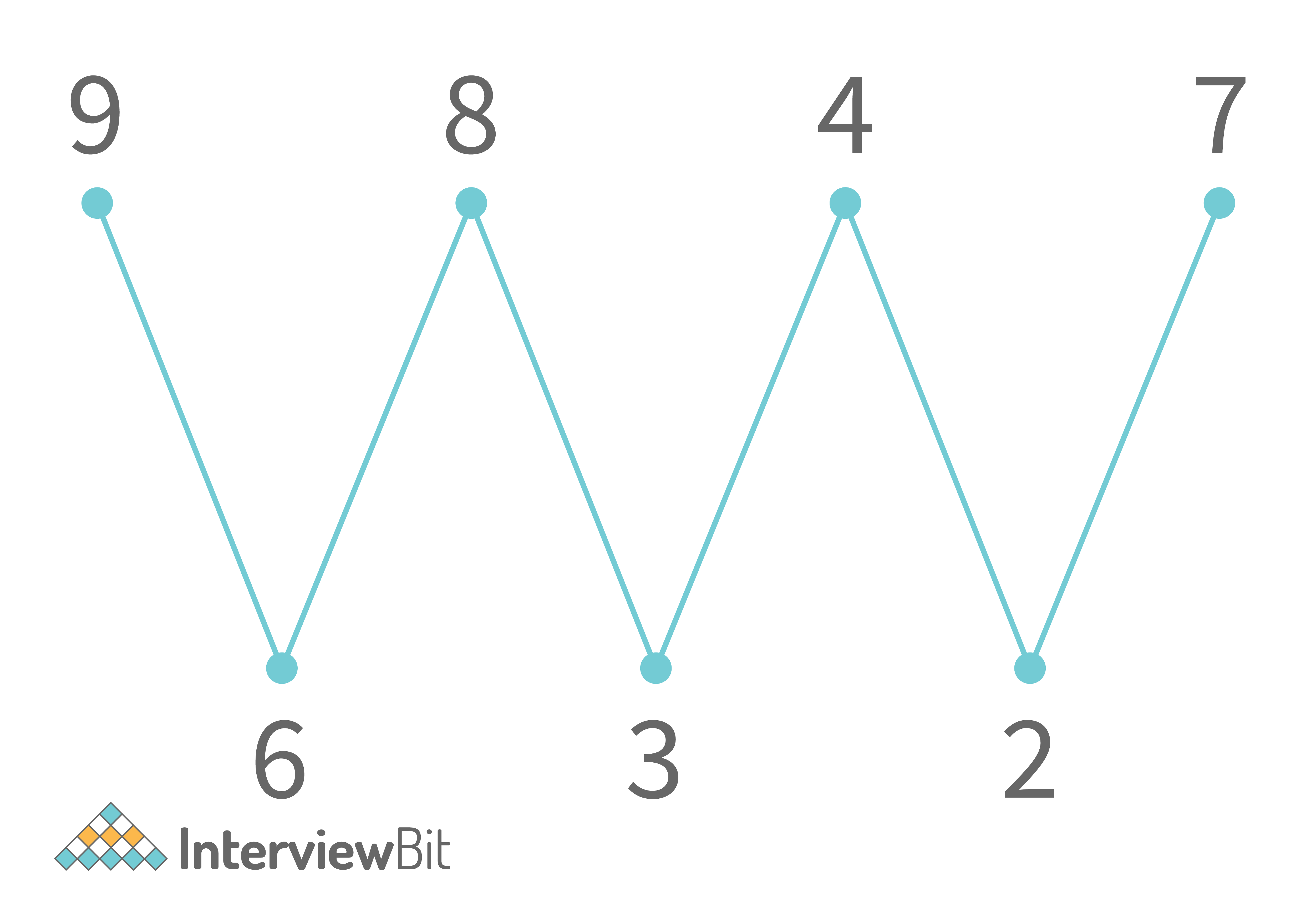
Output: 2

* **Time Complexity:**The time complexity is O(log2N) where N is the number of elements in the array.
* **Auxiliary Space:**O(1) as we have not used any extra space.

**5. You are given an array of integers. Your task is to Wave sort the array. For example, let us say the array is arr = {9,8,6,3,4,7,2}. Currently, the graph for the array of elements looks like this:**



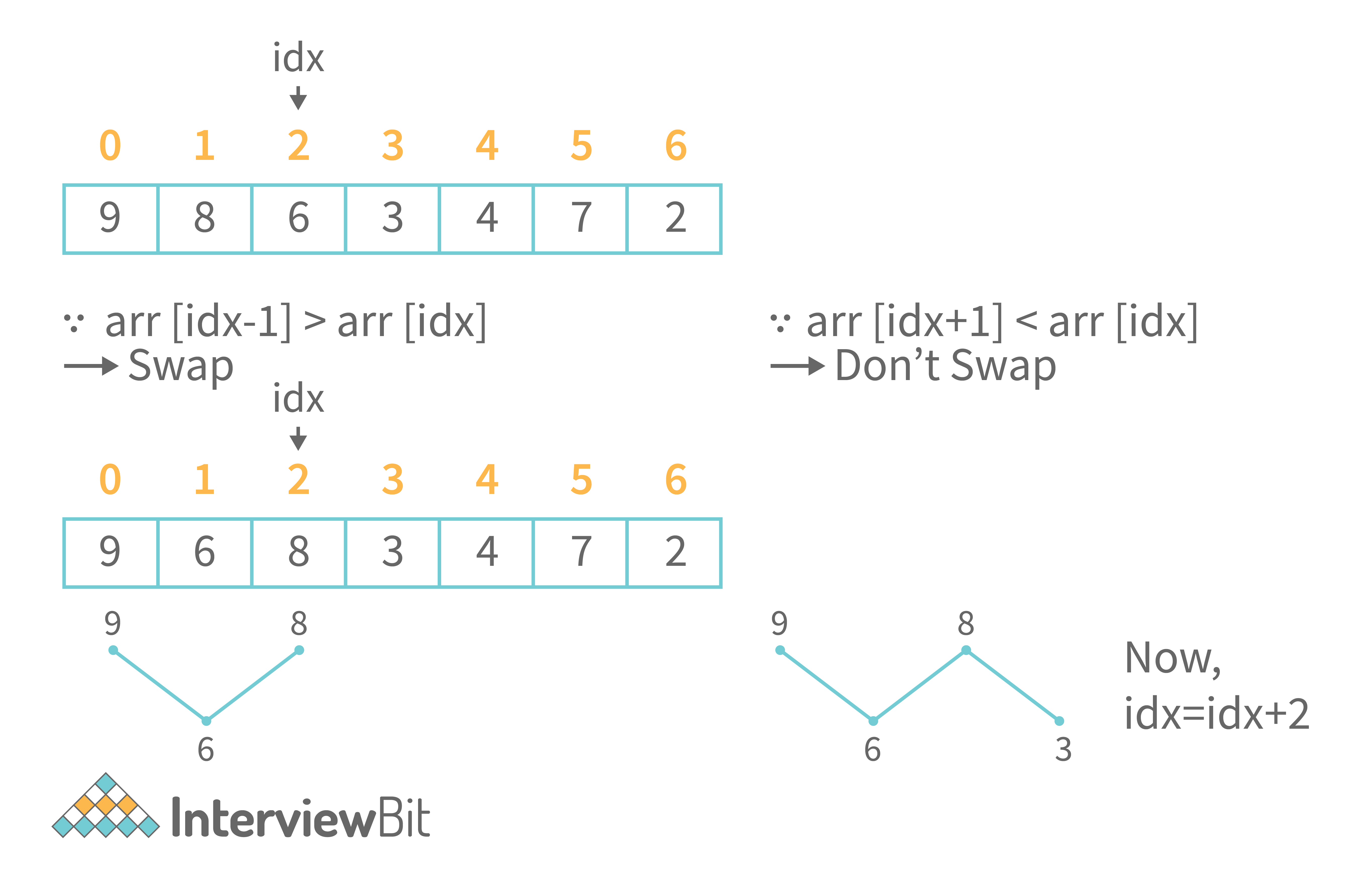
**We want the graph to look like this:**



**It is not necessary that you print the same order of elements as shown above. You can print any other order but the shape of the graph of elements of the array should look like a wave. The graph should always start with the peak and not a valley. You are not allowed to use any extra space and you have to solve this problem in O(N) time complexity.**

One basic approach to solve this problem is to sort the array and then swap the adjacent elements. The time complexity for which is O(NLog2N). Since we have to solve the problem in O(N) time complexity, we can solve it using the Wave Sort algorithm.

Our aim is to generate the Wave Graph. The aim can be accomplished by aiming at generating the peaks in the array or aiming at generating the valleys in the array. So, let us try to generate peaks in the array. Since we want the first array to be the peak, we will leave it as it is and start from the index = 2. Here, since we want to generate a peak, we need to have the previous and next elements smaller than our current elements. So, we will check that if our previous element is larger than our element, we will swap them. Again, at the same position, we will also check that the next element should be smaller than our current element. If it is not, swap these 2 elements. This is shown below.



So, we have to take a jump of 2 indices every time till we reach the end of the array. Hence, we will be able to wave sort the array in O(N) time and O(1) space.

**Java Code for Wave Sort**

**import** java.util.\*;

**class** **Main** {

**public** **static** **void** **swap**(**int**[] arr, **int** i, **int** j) {

**int** temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

**public** **static** **void** **waveSort**(**int**[] arr) {

**for**(**int** i=0;i<arr.length;i=i+2) {

**if**(i>0 && arr[i-1] > arr[i]) {

swap(arr,i-1,i);

}

**if**(i<arr.length-1 && arr[i+1] > arr[i]) {

swap(arr,i,i+1);

}

}

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**int** n = scn.nextInt();

**int**[] arr = **new** **int**[n];

**for**(**int** i=0;i<n;i++) {

arr[i] = scn.nextInt();

}

waveSort(arr);

System.out.println("After wave sort");

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

System.out.print(arr[i] + " ");

}

}

}

**Sample Output**

Input:

7

9 8 6 3 4 7 2

Output:

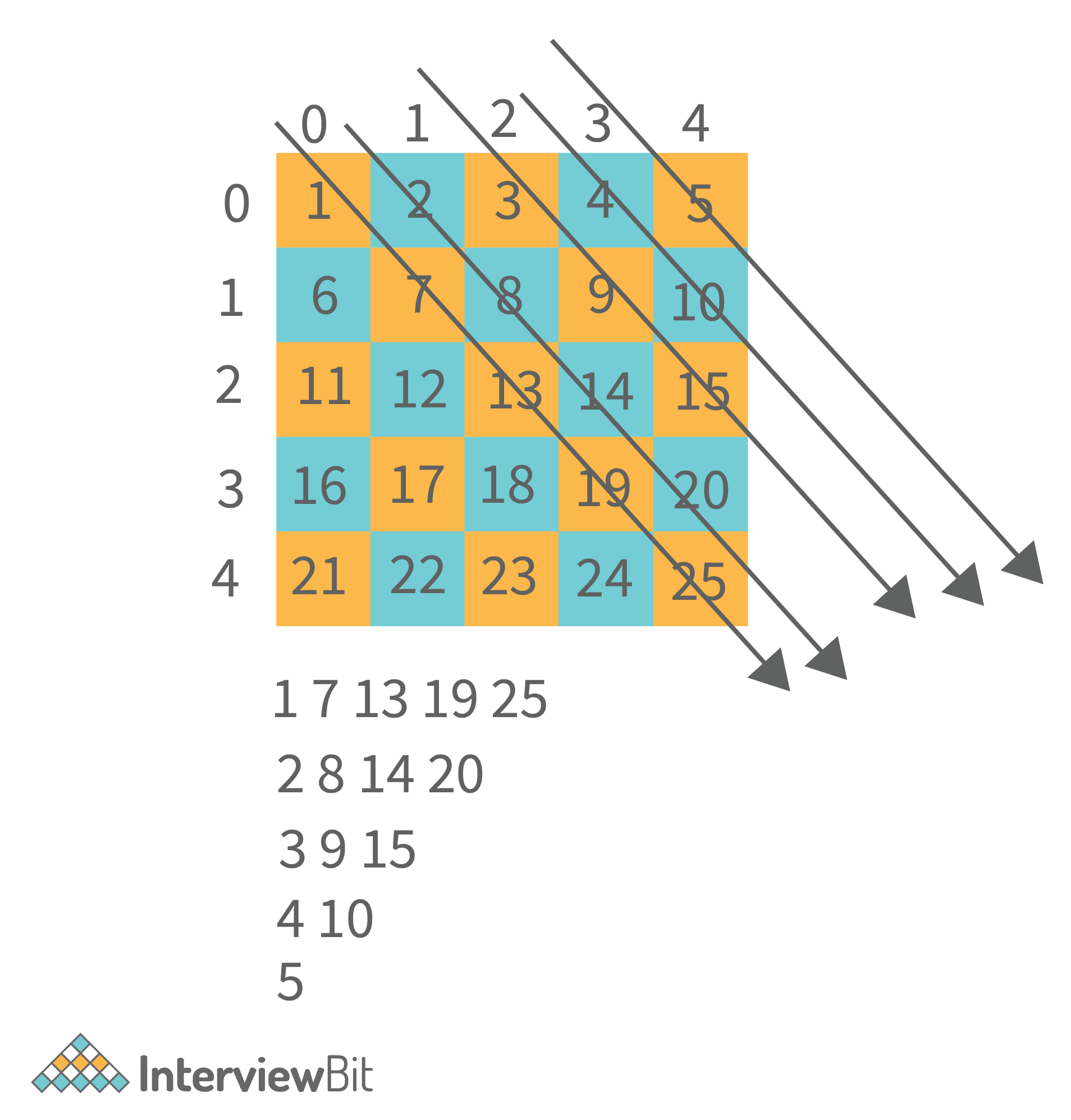
After wave sort

9 6 8 3 7 2 4

* **Corner Cases, You Might Miss:**Since we are swapping the previous element and the next element with the current element, we have to take care of the Index out of bounds condition. This is done in the code above.
* **Time Complexity:**O(N) as we are traversing the array.
* **Auxiliary Space:**O(1) as we have not used any extra space.

**6. You are given a 2-D array of size N x N. You have to print the elements of the array in diagonal order as shown below**

So, we have travelled the upper triangular half of the matrix diagonally. We can clearly see that the first diagonal has row = col i.e. the gap between them is 0. In the next diagonal, the column index is always greater than the row index by 1. The max gap up to which we can go is N-1, where N is the number of columns. So, we will use this gap strategy to traverse the matrix diagonally as shown below.



**Java Code for Diagonal Traversal**

**import** java.util.\*;

**public** **class** **Main** {

**public** **static** **void** **main**(String[] args) **throws** Exception {

// write your code here

Scanner scn = **new** Scanner(System.in);

**int** n = scn.nextInt();

**int**[][] mat = **new** **int**[n][n];

**for**(**int** i=0;i<n;i++) {

**for**(**int** j=0;j<n;j++) {

mat[i][j] = scn.nextInt();

}

}

diagonalTraversal(mat);

}

**public** **static** **void** **diagonalTraversal**(**int**[][] mat) {

**int** maxGap = mat[0].length - 1;

**for**(**int** gap=0;gap<=maxGap;gap++) {

**for**(**int** i=0,j=gap;i<mat.length && j<mat[0].length;i++,j++) {

System.out.print(mat[i][j] + " ");

}

System.out.println();

}

}

}

**Sample Output**

Input:

5

1  2  3  4  5

6  7  8  9  10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 25

Output:

1 7 13 19 25

2 8 14 20

3 9 15

4 10

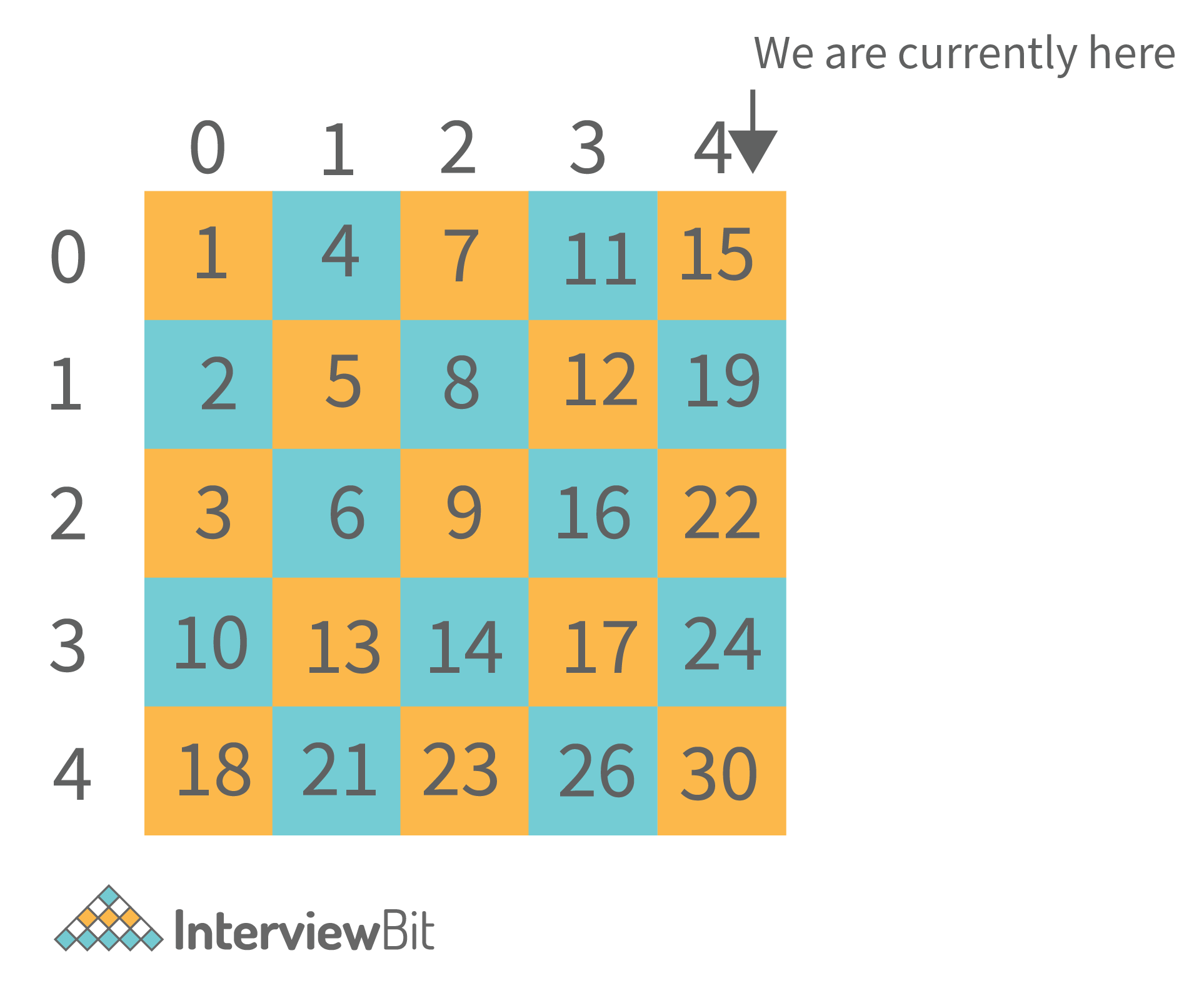
5

* **Time Complexity:**O(N2) as we have to traverse half matrix of size N x N.
* **Auxiliary Space:**O(1) is the auxiliary space.

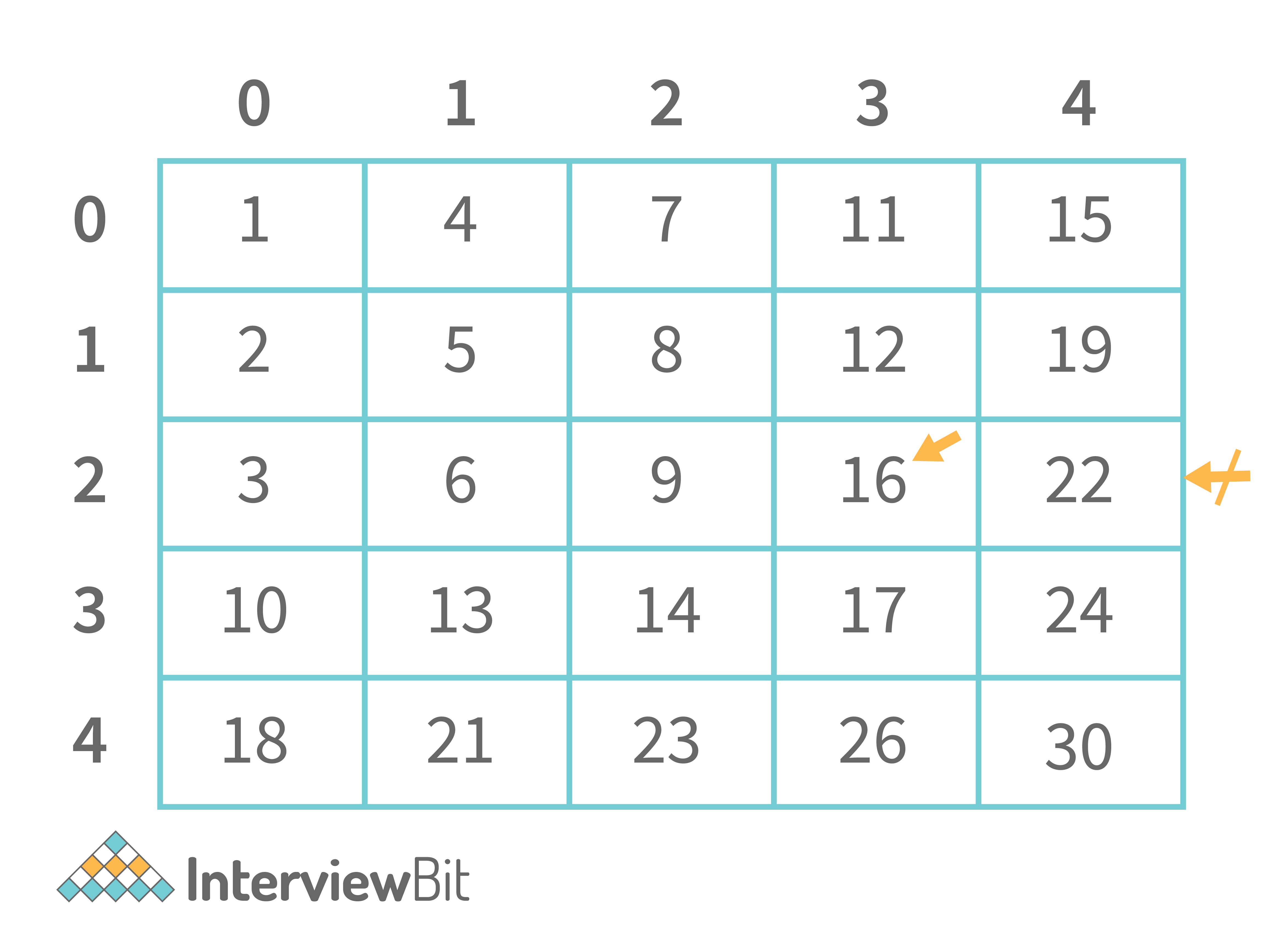
**7. Write a program in Java to search an element in a row-wise and column-wise sorted 2-D matrix of size M x N.**

**You have to search the element in O(N + M) time complexity without using any extra space. Print “true” if the element exists in the matrix else print “false”.**

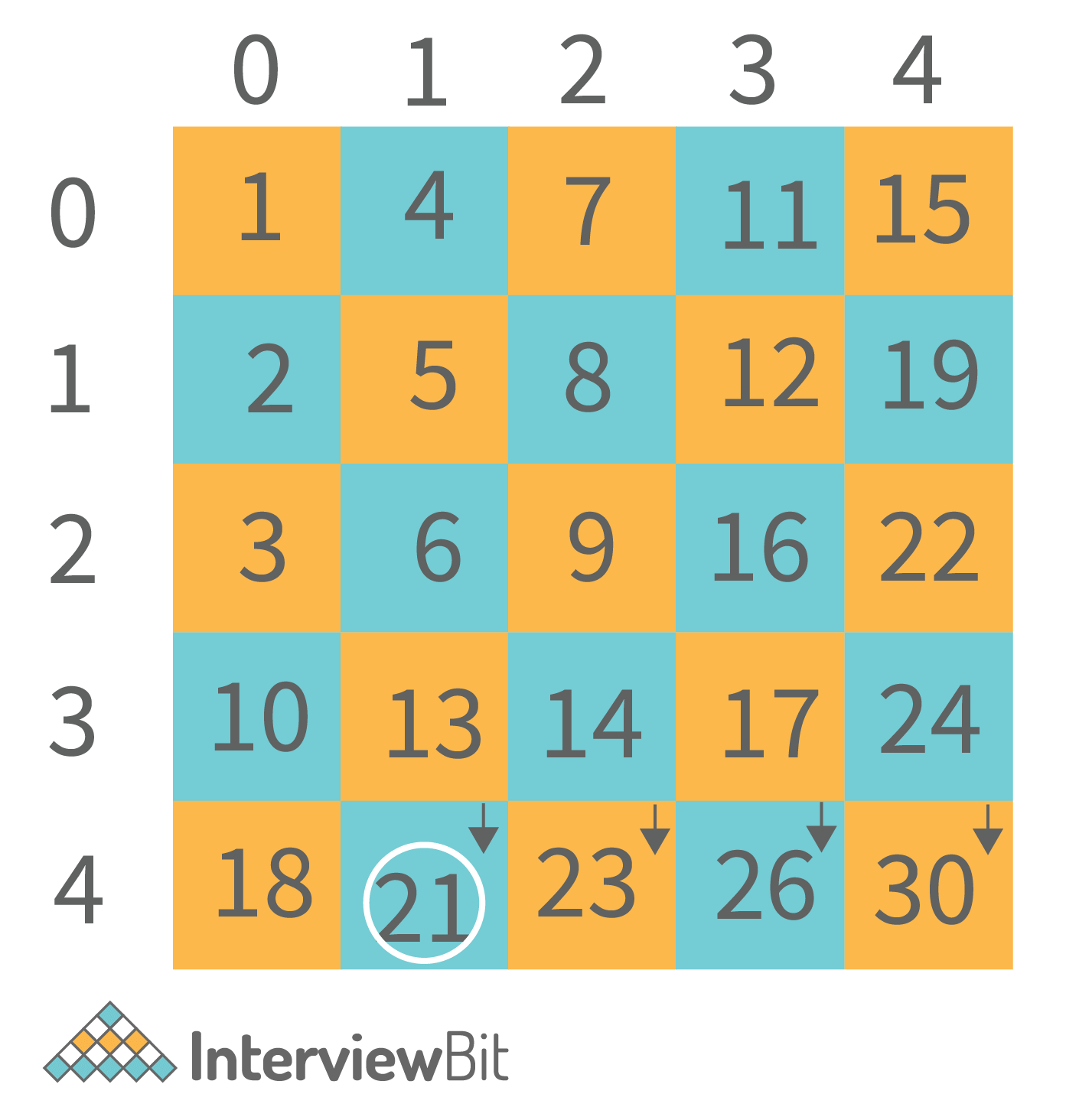
The normal searching technique will take O(N2) time complexity as we will search every element in the matrix and see if it matches our target or not. The other approach uses the fact that the elements are sorted row-wise. We can apply binary search on every row. Hence, the time complexity will be O(Nlog2N)  
Since we want the solution in O(N + M), this approach is not the one we will use. We will use the **Staircase Search Algorithm**. See, we know that the elements are sorted column-wise and row-wise. So, we start from the last element of the first row as shown below



Let us say we want to search for 21. We know that 21 is larger than 15. Since the matrix is row-wise sorted, element 15 is the largest element of this row. So, we are not going to find 21 in this row. So, we move directly to the last element of the next row. The Same is the case here as well. So, we move to the next row. Here, the element is 22. So, 21 might be present in this row. So, we move one step backwards in this row only.



On moving one step backwards, we see that we reach 16. Since this number is smaller than our target of 21, we know that we will not find our target in this row. Hence, we move to the last element of the next row and the same happens here too. Now, we are in the last row. We know that element might exist in this row. So, we keep on moving back in this row and find element 21.



So, we will implement this same algorithm. This is called staircase search.

**Java Code for Staircase Search**

**import** java.util.\*;

**class** **Main** {

**public** **static** **boolean** **staircaseSearch**(**int**[][] matrix, **int** target) {

**if**(matrix == **null** || matrix.length == 0 || matrix[0].length == 0) **return** **false**;

**int** j = matrix[0].length - 1 ;

**int** i = 0;

**while**(i < matrix.length && j>=0) {

**if**(matrix[i][j] == target) **return** **true**;

**else** **if**(matrix[i][j] < target) {

i++;

} **else** {

j--;

}

}

**return** **false**;

}

**public** **static** **void** **main**(String args[]) {

// Your code goes here

Scanner scn = **new** Scanner(System.in);

**int** N = scn.nextInt();

**int** M = scn.nextInt();

**int**[][] mat = **new** **int**[N][M];

**for**(**int** i=0;i<N;i++) {

**for**(**int** j=0;j<M;j++) {

mat[i][j] = scn.nextInt();

}

}

**int** target = scn.nextInt();

System.out.println(staircaseSearch(mat,target));

}

}

**Sample Output**

5 5

1  2  3  4  5

6  7  8  9  10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 25

21

Output: **true**

* **Time Complexity:**O(N + M) is the time complexity of staircase search. This is because we will have to search for a maximum of one row and one column.
* **Auxiliary Space:**O(1) as we have not used any extra space.

**8. Write a program in Java to create a user defined exception and also show it working means when it throws an exception.**

Here, you have to explain and write a user-defined exception of your own. This code is just for reference purposes. So, we are going to create an exception called LowBalanceException for a bank. So, whenever a person comes to the bank to create a bank account, the minimum account balance should be 5000. So, if the balance is less than 5000, the exception will be thrown. Let us write the code for the same.

**Java Code for User-Defined Exception**

**public** **class** **Main** {

**public** **static** **void** **main**(String[] args) {

Account a1 = **new** Account(500);

Account a2 = **new** Account();

a2.setBalance(500);

Account a3 = **new** Account(10000);

System.out.println("a1 balance = " + a1.getBalance() + " a2 balance = " + a2.getBalance() + " a3 balance = " + a3.getBalance());

}

}

**class** **Account** {

**private** **int** balance;

Account() {

balance = 5000;

}

Account(**int** balance) {

**try** {

**if**(balance>=5000) {

**this**.balance = balance;

System.out.println("The account is created and the balance is set to: "+ balance);

} **else** {

**this**.balance=0;

System.out.println("Account can not be created");

**throw** **new** LowBalanceException();

}

} **catch**(LowBalanceException e) {

System.out.println(e);

}

}

**void** **setBalance**(**int** balance) {

**try** {

**if**(balance>=5000) {

**this**.balance = balance;

System.out.println("The account is created and the balance is set to: "+ balance);

} **else** {

**this**.balance=0;

System.out.println("Account can not be created");

**throw** **new** LowBalanceException();

}

} **catch**(LowBalanceException e) {

System.out.println(e);

}

}

**int** **getBalance**() {

**return** balance;

}

}

**class** **LowBalanceException** **extends** **Exception** {

**public** String **toString**() {

**return** "Low Balance: The balance cannot be less than Rs.5000/-";

}

}

**Output**

The account can not be created

Low Balance: The balance cannot be less than Rs.5000/-

The account can not be created

Low Balance: The balance cannot be less than Rs.5000/-

The account is created and the balance is set to 10000

a1 balance = 0 a2 balance = 0 a3 balance =10000

**9. Write a program in Java to show multiple inheritance.**

Multiple inheritance is not possible in Java. So, we can use Interfaces in Java to create a scenario of multiple inheritance. In our example below, we have a class called Phone and a class called SmartPhone. We know that a SmartPhone is a Phone, however, it has various other features as well. For instance, a SmartPhone has a camera, a music player, etc. Notice that a SmartPhone **is a**Phone and **has a**camera and **has a**Music Player. So, there is one **is-A** relationship and multiple **has-A**relationships. The **is-A** relationship denotes extending the features and the **has-A**relationship denotes implementing the features. This means that a SmartPhone is a Phone i.e. it extends the features of a Phone however, it just implements the features of a Music Player and a Camera. It itself is not a music player or a camera. Following is the code for the above discussion.

**Java Code for Multiple Inheritance**

**public** **class** **Main** {

**public** **static** **void** **main**(String[] args) {

SmartPhone sp1 = **new** SmartPhone();

Phone p1 = **new** SmartPhone();

ICamera c1 = **new** SmartPhone();

IMusicplayer m1 = **new** SmartPhone();

sp1.videocall();

p1.call();

p1.message();

c1.click();

c1.record();

m1.play();

m1.pause();

m1.stop();

}

}

**class** **Phone** {

**void** **call**() {

System.out.println("call");

}

**void** **message**() {

System.out.println("Message");

}

}

**interface** **ICamera** {

**void** **click**();

**void** **record**();

}

**interface** **IMusicplayer** {

**void** **play**();

**void** **pause**();

**void** **stop**();

}

**class** **SmartPhone** **extends** **Phone** **implements** **ICamera**, **IMusicplayer** {

**void** **videocall**() {

System.out.println("Video call");

}

@Override

**public** **void** **click**() {

System.out.println("Picture click");

}

@Override

**public** **void** **record**() {

System.out.println("Record video");

}

@Override

**public** **void** **play**() {

System.out.println("Play music");

}

@Override

**public** **void** **pause**() {

System.out.println("Pause Music");

}

@Override

**public** **void** **stop**() {

System.out.println("Stop music");

}

}

**Output**

Video call

Call

Message

Picture click

Record Video

Play music

Pause Music

Stop music

**10. Write a program in Java, to show nesting of classes.**

Nesting of classes means writing a class inside another class. The inner classes in Java are usually static. This happens when we don’t want to use the variable of the outer class in the inner class. This helps to create an instance/object of the inner class without creating an instance of the Outer class. Following is a program to show the nesting of classes in Java.

**Java Code**

**public** **class** **Main** {

**public** **static** **void** **main**(String[] args) {

Outer obj1 = **new** Outer(10, 20);

Outer.Inner2 obj2 = **new** Outer.Inner2(40);

obj2.showData();

Outer.Inner3.z = 100;

System.out.println(Outer.Inner3.z);

}

}

**class** **Outer** {

**private** **int** x, y;

Outer() {

System.out.println("Outer class default constructor called");

}

Outer(**int** x, **int** y) {

**this**.x = x;

**this**.y = y;

}

**void** **showData**() {

System.out.println("the value of x is:" + x + " and the value of y is: " + y);

}

**class** **Inner1** {

**int** z = 0;

Inner1() {

System.out.println("Inner class default constructor called");

}

Inner1(**int** z) {

**this**.z = z;

}

**void** **showData**() {

System.out.println("The value of x is: " + x + " the value of y is: " + y + " and z is: " + z);

}

}

**static** **class** **Inner2** {

**int** z = 0;

Inner2() {

System.out.println("Inner class default constructor called");

}

Inner2(**int** z) {

**this**.z = z;

}

**void** **showData**() {

System.out.println("The value of z is: " + z);

}

}

**static** **class** **Inner3** {

**static** **int** z = 0;

Inner3() {

System.out.println("Inner class default constructor called");

}

Inner3(**int** a) {

z = a;

}

**void** **showData**() {

System.out.println("The value of z is: " + z);

}

}

}

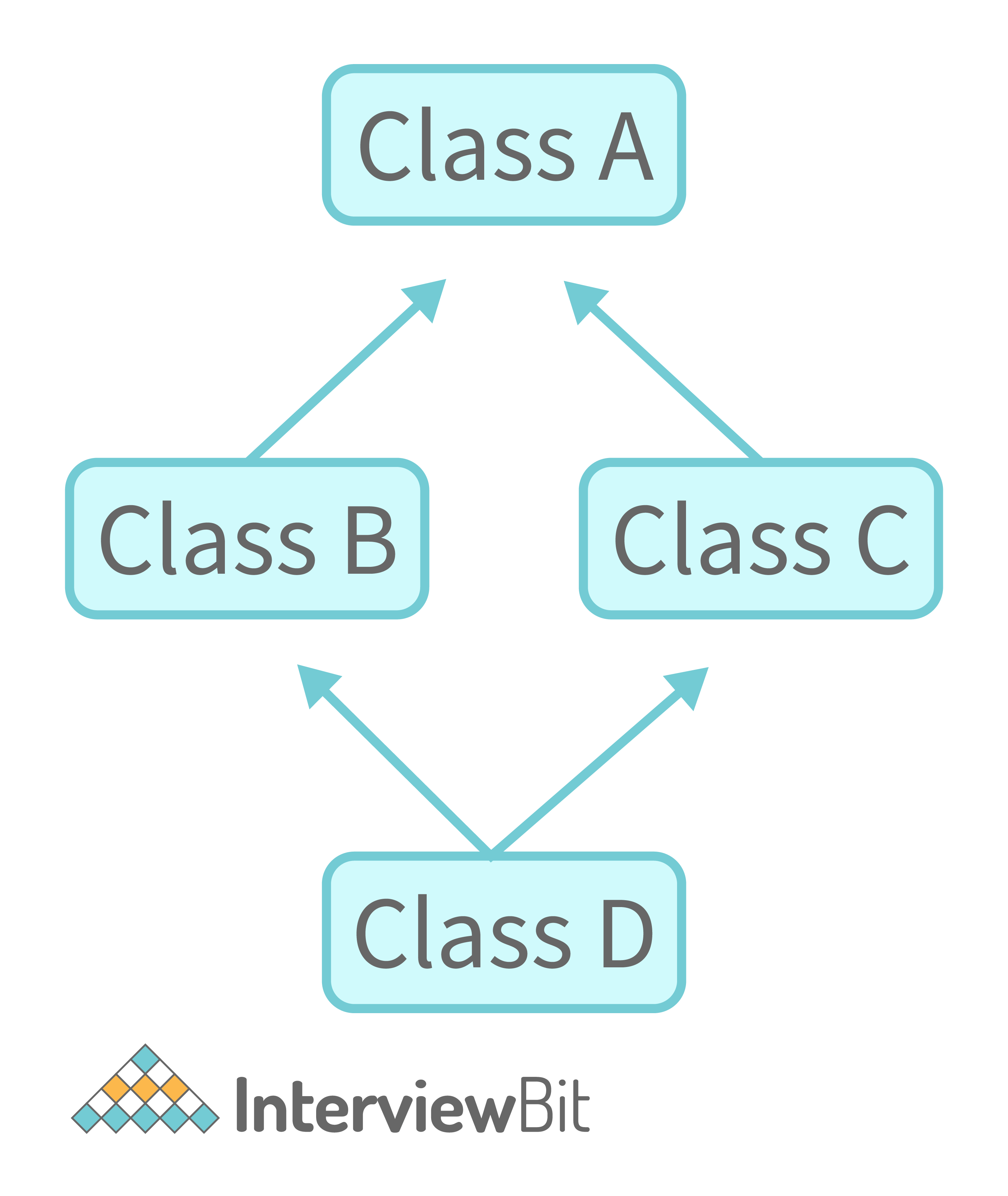
**Output**

The value of z is: 40

100

**11. Write a program in Java to show the Diamond Problem.**

The Diamond Problem is a problem of Multiple inheritance. It is one of the major reasons why multiple inheritance is not supported in Java. Have a look at the diagram given below.



Here, class D extends from classes B and C and they extend from Class A. Let us say that class A has a function called print(). This function is overridden in Class B and C respectively. Now, when class D extends B and C both, say it calls super.print(). Which function should be called? This is an anomaly called the **diamond problem** or **deadly** **diamond of death**.

**Java Code for Diamond Problem**

**class** **A** {

**public** **void** **print**() {

System.out.println("Class A print method");

}

}

**class** **B** **extends** **A** {

@Override

**public** **void** **print**() {

System.out.println("Class B print method");

}

}

**class** **C** **extends** **A** {

@Override

**public** **void** **print**() {

System.out.println("Class C print method");

}

}

//multiple inheritance not allowed in Java

**class** **D** **extends** **A**,**B** {

@Override

**public** **void** **print**() {

System.out.println("Class D print method");

}

}

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

D obj = **new** D();

obj.print();

}

}

**Output**

This compilation error occurs because multiple inheritance is not allowed in Java.

**12. Write a program in Java to show isAlive() and join() operations in multithreading.**

The isAlive() method tells whether a thread is alive or terminated. These alive and terminated are the states of a thread in Java. Also, the join() operation joins a thread to another. This means that the thread will wait for the complete execution of the thread to which it is joined even if its own work is completed. Then, they both will terminate together.

**Java Code to show isAlive() and join() operations**

**class** **DemoThread** **extends** **Thread** {

**public** **DemoThread**(String name) {

**super**(name);

setPriority(MAX\_PRIORITY);

}

}

**class** **DemoThread2** **extends** **Thread** {

**public** **void** **run**() {

**int** count = 1;

**while** (**true**) {

System.out.println(count);

count++;

**try** {

Thread.sleep(100);

} **catch** (InterruptedException e) {

System.out.println(e);

}

}

}

}

**public** **class** **Main** {

**public** **static** **void** **main**(String[] args) {

DemoThread t = **new** DemoThread("Thread 1");

System.out.println("ID " + t.getId());

System.out.println("NAME " + t.getName());

System.out.println("PRIORITY " + t.getPriority());

t.start();

System.out.println("STATE " + t.getState());

System.out.println("ALIVE " + t.isAlive());

DemoThread2 t2 = **new** DemoThread2();

**try** {

Thread.sleep(100);

} **catch** (Exception e) {

}

t2.setDaemon(**true**);

t2.start();

// t2.interrupt();

Thread mainThread = Thread.currentThread();

**try** {

mainThread.join(); // Now main will not terminate till the daemon thread is terminated

} **catch** (Exception e) {

}

}

}

**Output**

ID 13

NAME Thread 1

PRIORITY 10

STATE RUNNABLE

ALIVE **false**

1

2

3

4

5

6

7

**13. Write a program in Java to show Thread Synchronization.**

Here, you can create any 2 threads and synchronize them by using the **synchronized** keyword. An example program is given below.

**Java Program to show Thread Synchronization**

**class** **Table** {

**public** **synchronized** **void** **display**(**int** n) {

**for** (**int** i = 1; i <= 10; i++) {

System.out.println(n \* i);

}

}

}

**class** **Thread1** **extends** **Thread** {

Table t;

**public** **Thread1**(Table t) {

**this**.t = t;

}

**public** **void** **run**() {

t.display(5);

}

}

**class** **Thread2** **extends** **Thread** {

Table t;

**public** **Thread2**(Table t) {

**this**.t = t;

}

**public** **void** **run**() {

t.display(6);

}

}

**public** **class** **Main** {

**public** **static** **void** **main**(String[] args) {

Table table = **new** Table();

Thread1 th1 = **new** Thread1(table);

Thread2 th2 = **new** Thread2(table);

th1.start();

th2.start();

}

}

**Output**

5

10

15

20

25

30

35

40

45

50

6

12

18

24

30

36

42

48

54

60

**Additional Resources**

* [Java Cheat Sheet](https://www.interviewbit.com/java-cheat-sheet/)
* [Java Compiler to Practice](https://www.interviewbit.com/online-java-compiler/)
* [Java Interview Questions for 5 Years Experience](https://www.interviewbit.com/java-interview-questions-for-5-years-experience/)
* [All Technical Interview Questions](https://www.interviewbit.com/technical-interview-questions/)
* [Java vs Python](https://www.interviewbit.com/blog/java-vs-python/)
* [Java Projects](https://www.interviewbit.com/blog/java-projects/)
* [Java Developer Salary](https://www.interviewbit.com/blog/java-developer-salary/)

**Java Programming MCQ**

1.

**Which of the following snippets of code is correct to check the equality of strings in Java?**

str1 == str2

equals(str1,str2)

str1.equal(str2)

str1.equals(str2)

2.

**Which of the following will create an ArrayList of Integers in Java?**

ArrayList<int> list = new ArrayList<>();

List<int> list = new ArrayList<>();

List<Integer> list = new ArrayList<>();

ArrayList<Integer> list = new List<>();

3.

**What is the output of the following Java program?**

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

String s1 = "InterviewBit";

String s2 = **new** String("InterviewBit");

System.out.println(s1 == s2);

System.out.println(s1.equals(s2));

}

}

false true

true true

true false

false false

4.

**What is wrong with this program?**

**class** **A** {

**public** **void** **print**() {

System.out.println("Class A print method");

}

}

**class** **B** **extends** **A** {

@Override

**public** **void** **print**() {

System.out.println("Class B print method");

}

}

**class** **C** **extends** **A** {

@Override

**public** **void** **print**() {

System.out.println("Class C print method");

}

}

**class** **D** **extends** **A**,**B** {

@Override

**public** **void** **print**() {

System.out.println("Class D print method");

}

}

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

D obj = **new** D();

obj.print();

}

}

Multiple inheritance is not allowed in Java.

Diamond Problem

None of the above

Both A and C

5.

**What is the output of the following code?**

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

**final** **int** x = 10;

**int** y = x++;

System.out.println(y);

}

}

10

Compilation Error

Runtime Error

11

6.

**What is the output of the following code?**

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

**int**[] arr = {9,8,4,2,1,5,6,7,2,6,3,9,2,2,1};

HashSet<Integer> hs = **new** HashSet<>();

**for**(**int** x : arr) {

hs.add(x);

}

**for**(**int** x : hs) {

System.out.print(x + " ");

}

}

}

9 8 4 2 1 5 6 7 3

9 8 4 2 1 5 6 7 2 6 3 9 2 2 1

4 8 2 1 9 6 5 7 3

Either A or C

7.

**What is the output of the following code?**

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

**int** x = 10/0;

System.out.println(x);

System.out.println("end of program");

}

}

Arithmetic Exception occurs and the program stops

Arithmetic Exception is thrown and “end of program” is printed after the exception is thrown.

Arithmetic Exception is thrownbut “end of program” is printed before the exception is thrown.

Only “end of program” is printed and no exception occurs.

8.

**What is the output of the following program?**

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

**try** {

**int** x = 10/0;

System.out.println(x);

} **catch**(Exception e) {

}

System.out.println("end of program");

}

}

Arithmetic Exception occurs and the program stops

Arithmetic Exception is thrown and “end of program” is printed after the exception is thrown.

Arithmetic Exception is thrown but “end of program” is printed before the exception is thrown.

Only “end of program” is printed and no exception is thrown.

9.

**Which of the following is correct according to the program given below?**

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

**int** a = 10;

**int** b = 20;

swap(a,b);

}

**public** **static** **void** **swap**(**int** a, **int** b) {

**int** temp = a;

a = b;

b = temp;

}

}

a = 10 b=10

a = 10 b=20

a = 20 b=20

a = 20 b=10

10.

**What is the output of the following code?**

**class** **Main** {

**public** **static** **void** **main**(String args[]) {

// Your code goes here

**int**[] arr = {10,20};

swap(arr);

}

**public** **static** **void** **swap**(**int**[] arr) {

**int** temp = arr[0];

arr[0] = arr[1];

arr[1] = temp;

}

}

arr[0] = 10 arr[1] = 10

arr[0] = 10 arr[1] = 20

arr[0] = 20 arr[1] = 10

arr[0] = 20 arr[1] = 20