



Search



2021: No salary increase
2022: No salary increase
2023: No salary increase

...see more

5,970

164 comments • 166 reposts



Like



Comment



Repost



Send



Japneet Sachdeva celebrates this

...



Alex Siminiuc • 3rd+
SDET at Allegis Group
14h •

[+ Follow](#)

Selenium tests fail a lot.

Oh no!!!

Selenium sucks, lets change it with

Wait a moment.

The Selenium test may fail also because:

- bad automation code
- dependent tests
- long test cases
- bad test cases
- dynamic HTML
- unstable site
- site shared with dev team and many other QA teams
- slow environment
- changing site
- unstable grid

Let's investigate all these first before blaming Selenium.

369

15 comments • 12 reposts



Like



Comment



Repost



Send

Job search smarter

...

Daini Sampath Kumar - Premium



Home



My Network



Post



Notifications



Notifications



Jobs



Search



Japneet Sachdeva celebrates this

...

X

**Isha Chaudhary** · 3rd+

Automation Engineer at KatchinTech | Quality Assurance expert | Selenium | TestNG

18h ·

[+ Follow](#)

Day 3: Selenium Automation Interview Questions

Question: What Test Automation Selenium Framework? Or Explain Test Automation Selenium Framework?

1. Definition: A Selenium test automation framework is a structured approach to automate web application testing efficiently. It combines tools, best practices, and design patterns to create effective and maintainable test scripts.

2. Key Components:

Programming Language: Java + Selenium – The main language for writing test scripts. Java is commonly used due to its industry-wide acceptance.

IDE: Eclipse – A popular Java IDE for coding and managing your test scripts.

Testing Framework: TestNG – Manages test execution, provides features like test grouping, parallel execution, and reporting.

Version Control: GIT – Tracks changes in test scripts and supports team collaboration.

Continuous Integration: Jenkins – Automates test execution and integrates with tools like Maven for continuous integration.

3. Key Features:

Page Object Model (POM): Separates page-specific logic into different classes for maintainability.

Test Base Class: Centralizes WebDriver setup and configuration.

Utility Class: Contains reusable functions like taking screenshots.

Property Files: Stores configurations like browser type and URLs.

Test Data: Managed through Excel sheets and Apache POI for data-driven testing.

TestNG: Organizes and runs tests, supports parallel and grouped test executions.

Maven: Manages project dependencies and builds.

Jenkins: Automates test runs and integrates with build



Home



My Network



Post



Notifications



Jobs

Industry-wide acceptance.

IDE: Eclipse – A popular Java IDE for coding and managing your test scripts.

Testing Framework: TestNG – Manages test execution, provides features like test grouping, parallel execution, and reporting.

Version Control: GIT – Tracks changes in test scripts and supports team collaboration.

Continuous Integration: Jenkins – Automates test execution and integrates with tools like Maven for continuous integration.

3. Key Features:

Page Object Model (POM): Separates page-specific logic into different classes for maintainability.

Test Base Class: Centralizes WebDriver setup and configuration.

Utility Class: Contains reusable functions like taking screenshots.

Property Files: Stores configurations like browser type and URLs.

Test Data: Managed through Excel sheets and Apache POI for data-driven testing.

TestNG: Organizes and runs tests, supports parallel and grouped test executions.

Maven: Manages project dependencies and builds.

Jenkins: Automates test runs and integrates with build pipelines.

Sample Answer

"Our Selenium framework uses Java for scripting and TestNG as the testing framework. We manage dependencies with Maven and use GitHub for version control. The framework is designed with a Data Driven approach and Page Object Model for maintainability. We read test data from Excel using Apache POI and generate detailed reports with Extent Reports. For logging, we use Log4j2 and capture screenshots for failed tests. We follow the DRY Principle to avoid redundancy, create page classes for common components, and use TestNG XML files to manage Test Suites like Sanity and Regression. Jenkins handles test execution and schedules the test suites based on their runtime, e.g., Sanity Suite runs daily and Regression Suites run nightly or weekly."

Feel free to share your tips or ask more questions in the comments!

#Selenium #AutomationTesting #InterviewTips



Home



My Network



Post



Notifications



Jobs



Celigo Interview Questions for #QA role (Candidate Ex: 5 yrs):

1. What is the difference between verification and validation?
2. How to write test cases?
3. What is the traceability matrix?
4. How to raise a defect
5. Have you written any test cases without documents? If required, then how will you write?
6. What is an interface in Java?
7. What is inheritance in Java?
8. How to implement an interface
9. How to extend a class
10. Is it possible to extend the two classes together?
11. Why do we use an interface?
12. What is XPath, & What are the different types of XPath?
13. Explain the assert and verify.
14. What are the types of joins in SQL?
15. How will you fetch data from excel?
16. Have you worked on TestNG?
17. What are the annotations?
18. Explain the annotations
19. What is the POM model?
20. Which locator do you prefer?
21. Explain both (relative and absolute) x-path
22. Webdriver API
23. How are you triggering your suite using Jenkins?
24. Have you used javascript in your code-if yes, where used in the code?



498



2

**softwaretestingandautomation**[Follow](#)

525

Save & Share | Celigo Interview Questions for #QA role - 5 years of experience 🐞 💬 ...



DJ GROSSU · Nature of Love



Bangalore, India



CSS Generic Locating & construct these with any X PATH (7)
 Console `$(` `)` HTML Web Elements `$x(` `)`
`tagname.className locator` `tagname ` is a link
`tagname#id locator` `(anchor a)`

Index [] → when there are same attributes to traverse b/w them
 we use today

`//Tagname[@attribute = 'value'] : nth-child(index)` `//Tagname[@attribute = 'value'][index]`

Customized

`Tagname[@attribute = 'value']` `//Tagname[@attribute = 'value']`

Regular expression - ~~if~~ if a attribute value has static + dynamic values (ex: Submit) then we use ~~part~~ part of value that is static. we use static value to find that element

`input[type*='pass']`
 ↓
 tagname attribute value (static)

`//button[contains(@class, 'Submit')]`
 ↓
 tagname attribute attribute value (static)

parent tagname to child tagname traversing

parent tagname child tagname

parent tagname / child tagname

Example for Xpath using all
 ↗ customized

`//div[@class = 'forget-pwd-link-container']/button[1]`
 ↓ ↓ ↓
 parent tagname attribute attribute value child tagname ↗ Index

→ traversing parent tagname to child tagname

→ customized path to identify the parent

→ Index to identify a child tagname because they have same tagnames

CSS
using Tagname
→ Tagname

X Path (8)

→ // tagname

→ Identifying element using text
(Ex - Log Out button)

// button [text () = 'Log Out']
↓
tagname ↓
text on the button

// div[@id='glschool_main_content_area_destination']
↓
tagname attribute ↓
value

// a[@value='MAA']
↓
tagname attribute value

// button [text () = 'Log Out']
↓
tagname attribute value

one space tagname attribute value stands for any tagname
if it is unique u can use *

Absolute Xpath :- means starting from the top, main parent, root object to travelling to your child element (eg - /html) (single slash represents starting from root of HTML tagname)

Relative Xpath :- means jumping into the middle not from starting (eg - /header) (double slash represents rel Xpath)
tagname ↓
rel Xpath #page → Automation Practice/

parent to child → // Header / div / button []

↓ index

parent to child & between siblings or children

→ // Header / div / button [] / following-sibling : button []
↓ parent to child ↓ between siblings

child to parent

→ // header / div / button [] / parent :: div
 ↓ one of middle parents

Interview ques Child → parent → its parent → another sibling

→ // header / div / button [] / parent :: div / parent :: header / a []
 ↓ child ↓ parent ↓ its parent ↓ another sibling

Implicit wait :- something we are searching for a element in a page

If it doesn't exist at that time and shows up at later time

then selenium waits until it shows up

Thread Sleep :- Something element that is already there & taking time to get stable (Design movements) in a single page

TestNG Tutorial

Importance of TestNG

Installation and configuration in Eclipse

Testng XML file Importance

Controlling Testcase executions from Testng XML files

Include Exclude Mechanism

Understanding TestNG Annotations

Groups

Data Provider Annotation

Annotation helper Attributes

Parameterization in TestNG xml file

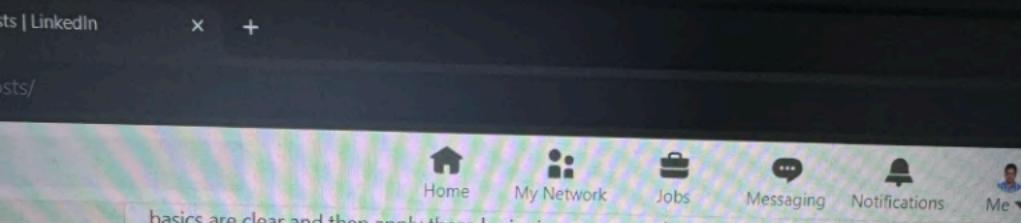
Running Tests in parallel with TestNG

Design Selenium Framework from Scratch

- Why Framework? Its advantages
- Create **Maven** Structured Framework with all necessary Automation dependencies
- Select Sample Ecommerce Application to Automate the end-to-end flow
- Implement **Page object Model** mechanism to drive the locators from respective classes
- Drive object creation within Page object classes **encapsulating** it from Tests
- Create Base Test which sets browser configuration details and **Global properties**
- Decide the **Test Strategy**, how tests should be clubbed & distributed with appropriate annotations
- Create **TestNG runner** file to trigger the tests with one Single point of execution control
- Introduce **Grouping in TestNG.xml** to categorize tests with different tags of execution
- Implement **Data driven testing & Parameterization** using TestNG Data provider HashMap & Json File readers
- Implement **TestNG Listeners** to capture Screenshot on automatic test failures and logging
- Create Extent Report wrapper to generate **excellent HTML reports** for the application
- Make Framework necessary changes to support **parallel execution** with

- Make Framework necessary changes to support parallel execution with Thread safe mechanism
- Implement TestNG retry mechanism to rerun the failed flaky tests in the application
- Run the Framework tests with Maven commands with TestNG Maven integration plugin
- Implement Maven Run time variables to replace global parameters of test data at run time
- Integrate the Framework with Jenkins with Parameterized Build Pipeline Jobs & Schedule the jobs on specific time frames
- Add Cucumber Wrapper to existing framework with Cucumber TestNG Runner

- Create **Feature files & Step definitions** to support Cucumber execution of Selenium Tests
- Understand how **cucumber tags, Data driven & Parameterization** works in running the tests.



basics are clear and then apply these logics in your general programming practices.

For each programming question I have attached video lecture provided by : Pavan Kumar Bhimavarapu and Naveen Khunteta

Note: These questions can be used to improve your programming logic, which helps in on the spot coding rounds!

- 1) Reverse a String = <https://bit.ly/3Qa34bn>
- 2) Generate Fibonacci series = <https://bit.ly/3Z7k5Y7>
- 3) Find largest and smallest in an array = <https://bit.ly/3Z1TrA3>
- 4) Search element in an array = <https://bit.ly/3G8LmS>
- 5) Sort elements in an array = <https://bit.ly/3i8b6VS>
- 6) How to remove unwanted characters from string = <https://bit.ly/3lcqs6m>
- 7) How to reverse each word = <https://bit.ly/3WD4y0w>
- 8) Find given substring is presence = <https://bit.ly/3G6w5jT>
- 9) How to find duplicates = <https://bit.ly/3i5usuD>
- 10) Static keyword questions = <https://bit.ly/3Z5kVEz>

Note = Do add up any other important interview programming questions in comment section below so that it can help other aspirants as well :)

Note - After solving 2-3 problems then try to solve the questions first on a piece of paper and then match your logic with videos.

-X-X-

Java Test Automation with Coding Practice: <https://lnkd.in/g5hr9bea>

Become SDET and future SDET Manager: <https://lnkd.in/gusymgFi>

#japneetsachdeva



Home



My Network



Jobs



Messaging



Notifications



Sapna Jatlieva · 4d ago

Sr. SDET | Top 1% Mentor & 5000+ sessions TopMate| Data ...
4d ·

...

Before jumping to LeetCode, HackerRank or CodeChef

Solve these questions without using the solutions!

find the frequency of each element in the array

print the duplicate elements of an array

print the elements of an array in reverse order

print the largest element in an array

print the smallest element in an array

print the sum of all the items of the array

sort the elements of an array in ascending order

find Second Largest Number in an Array

find Second Smallest Number in an Array

print Odd and Even Numbers from an Array

remove all the white spaces in a string



Write a code in Java to prove that String objects are immutable

program to count the number of words in a string

program to check whether a string is a Palindrome

Reverse a string

Remove Leading zeros

Print first letter of each word in a string

Given a string s, find the length of the longest substring without repeating

characters

-X-X-

To get solutions for above questions and learn Java use: <https://lnkd.in/g5hr9bea>

Become SDET & Future SDET Manager + 950+ Q&A Bank, Hero in Gen AI, Job techniques, Interview Prep Q&A Bank: <https://lnkd.in/gusymgfi>



Search





Home



My Network



Jobs



Messaging



Notifications

Me

||Java Most Imp Coding Interview Questions ✓

#SDET

1. Reverse a String:

Write a Java program to reverse a given string.

2. Find the Largest Element in an Array:

Find and print the largest element in an array.

3. Check for Palindrome:

Determine if a given string is a palindrome (reads the same backward as forward).

4. Factorial Calculation:

Write a function to calculate the factorial of a number.

5. Fibonacci Series:

Generate the first n numbers in the Fibonacci sequence.

6. Check for Prime Number:

Write a program to check if a given number is prime.

7. String Anagrams:

Determine if two strings are anagrams of each other.

8. Array Sorting:

Implement sorting algorithms like bubble sort, merge sort, or quicksort.

9. Binary Search:

Implement a binary search algorithm to find an element in a sorted array.

10. Duplicate Elements in an Array:

Find and print duplicate elements in an array.

11. Linked List Reversal:

Reverse a singly-linked list.

12. Matrix Operations:

Perform matrix operations like addition, multiplication, or transpose.



12. Matrix Operations:

Perform matrix operations like addition, multiplication, or transpose.

13. Implement a Stack:

Create a stack data structure and implement basic operations (push, pop).

14. Implement a Queue:

Create a queue data structure and implement basic operations (enqueue, dequeue).

15. Inheritance and Polymorphism:

Implement a class hierarchy with inheritance and demonstrate polymorphism.

16. Exception Handling:

Write code that demonstrates the use of try-catch blocks to handle exceptions.

17. File I/O:

Read from and write to a file using Java's file I/O capabilities.

18. Multithreading:

Create a simple multithreaded program and demonstrate thread synchronization.

19. Lambda Expressions:

Use lambda expressions to implement functional interfaces.

20. Recursive Algorithms:

Solve a problem using recursion, such as computing the factorial or Fibonacci sequence.

Credits: Pratima Singh

Follow Kushal Parikh for insightful content on Software Automation Testing!

 Q&A on Automation Testing With Selenium (Freshers):

<https://lnkd.in/gtNxEV5v>

Like, Follow & Share it with your friends. 

Selenium Framework Interview Questions:

1. What is the Design Pattern you have used in writing the tests in the Framework?
2. How are reusable utilities handled within the framework?
3. Where did you use Inheritance OOPS Concept in your framework?
4. How did you drive the data from external files in the framework?
5. Did you use Interfaces in the framework? If so, what is the scope of it?
6. How are you achieving Encapsulation in the framework?
7. Does your Framework Support Parallel Runs? How are you writing Thread safe code?
8. Do you have static keywords in the framework? If so, its usage?
9. How are you sending Global Properties to your test at run time?
10. What is the mechanism you use to run only selected set of tests inside the framework?
11. How are you handling Flaky tests in the framework?
12. Does your Framework take Screenshot on test

Work Interview Questions - Part 2

Design Selenium Framework from Scratch

Home Insert References Mailings Review View Tell me

1. What is the Design pattern you have used in writing the tests in the Framework?
2. How are reusable utilities handled within the framework?
3. Where did you use Inheritance OOPS Concept in your framework?
4. How did you drive the data from external files in the framework?
5. Did you use Interfaces in the framework? If so, what is the scope of it?
6. How are you achieving Encapsulation in the framework?
7. Does your Framework Support Parallel Runs? How are you writing Thread safe code?
8. Do you have static keywords in the framework? If so, its usage?
9. How are you sending Global Properties to your test at run time?
10. What is the mechanism you use to run only selected set of tests inside the framework?
11. How are you handling Flaky tests in the framework?
12. Does your Framework take Screenshot on test failures? How did you implement it?
13. Explain Framework Architecture

Selenium Framework Interview Questions

1. Design pattern

We used Page Object Model. each & every time present in the SOT is derived from the page object classes. And also used Page Factory pattern to create objects. Page Objects are only for elements and action methods. The Interpolated objects into action methods.

2. How are reusable utilities handled within the framework?

A. Created one class Abstract Components for reusing common methods. Page object classes extends Abstract Components. We also have another reusable methods from test perspective created Base Test. Test + the classes consisting Test methods extends Base Test. Test related reusable methods like initializing driver, launching App, tear down.

3. usage of Inheritance (OOPS concept) in your framework

A. we have some reusability classes. we are simply inheriting to child classes

4. How do you derive the data from external files in the framework

A. we used json files to parse the data. In json file, no of entries created so the test runs that many no of times. so the test is parameterized with 2 diff data sets and runs 2 times using Data Providers in TestNG using Hashmap as a object

5. Did you use Interfaces in the framework? If so, what's the scope?

A. yes we used that in listeners. Listener class implements ^{Interface} ~~ITestListener~~. we implemented this interface bcoz it provides intelligent methods like example taking screenshot when test method dry failed & also WebDriver is a Interface & ChromeDriver is a Class

6. How are you achieving Encapsulation in ~~the~~ the framework

A. Hiding the implementation details of your class from another class. suppose we should call only action methods into this class from other class & we shouldn't call webElements into this class from other class. then we make that WebElements as private then they are strictly accessible within the class

7. Does your framework support parallel runs? How are you writing thread safe code?

A. In testing.xml file put parallel = "tests" beside suite. Tests will run fine with its own thread & coming to reporting the variables may override. we can solve this override using Thread local class. (unique Thread id $SOT \rightarrow$ test)

8. Do you have static keywords in the framework? If so, what's the usage?

A. the danger with static is if you mark them as static then variables will be shared across the threads if we run two tests parallel here same defined variable will be shared among two threads of tests. parallel testing comes into picture there there is no scope for static keywords

when two tests run in parallel ^{in testing.xml} ~~two threads are~~ ③
two tests has separate threads, when those tests when 1st
thread accessing the driver it creates fresh instances and variables
and the same another thread accessing the driver parallelly it
again creates fresh instance and variables for its own thread
no colliding. if we mark them as static then variable
will be shared across the threads & ~~if~~ now if we run them
parallelly then ^{same} driver variable will be shared for two threads if
we mark them as static (common space allocation for all the threads)
parallel testing → no static keywords XXX. we can use static ^{in parallel}
tests which are independent of your test runs examples like properties
files (read only not any code written). the advantage of static is like marking
prop objects static ^(not for) every thread in parallel runs we are creating
prop objects for thread 1 & thread 2 ^{but now} we can save space by making
them static so one object is created for all threads. object is
shared among all the threads.

Q. How are you sending global properties at run time

A. 1. Creating properties file (GlobalData.prop) → browser=chrome reading these properties
into Base Test Class initialize driver method using Properties Class (.prop extension)
loading that file & reading it at run time. Global properties like
examples → (browser=chrome, edge,firefox), url (QA, UAT, Prod, DEV environments)
set implicit timeouts

2. we can also drive the global properties at run time using
maven terminals (-P, -D) 3. Jenkins

10. what is the mechanism you use to run only selected set of tests inside the framework (4)
- A. Using TestNG groups concepts. groups help attribute to the testNG annotations, only run those tests which are identified as smoke in XML file by writing <groups> </groups> and also in pom.xml by creating profile, and run those tests using maven command
 a profile give that separately created
 group and groups XML

11. How are handling flaky tests in the Framework?

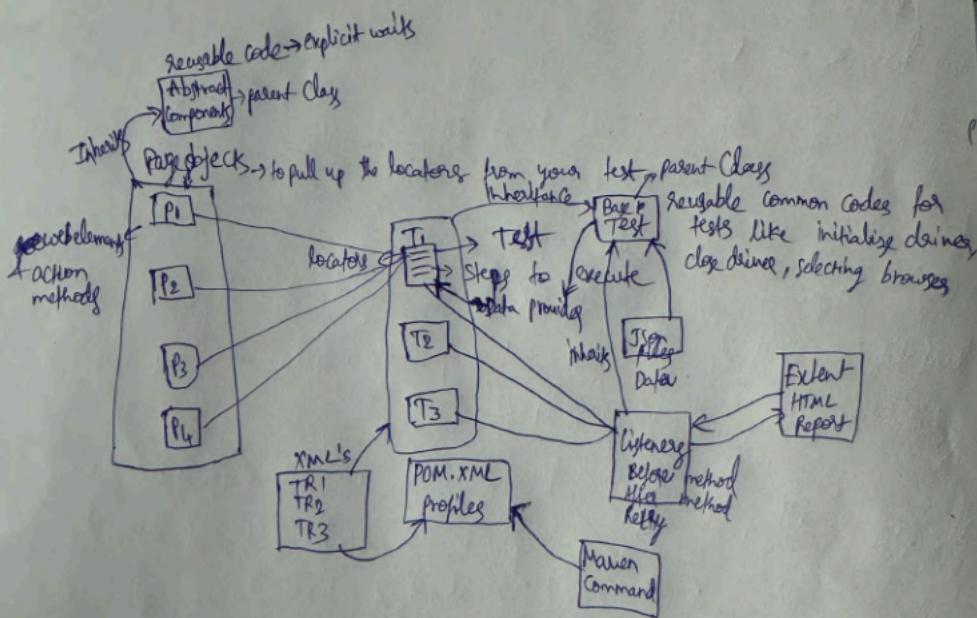
A. Flaky tests means tests that are inconsistent & might fail randomly but passing the 2nd attempt. Retry implements FlakyAnalyzer Interface (testNG interface) + whenever test fails testNG comes to this method (retry) if it returns true then test will retrying it continuously until this methods returning false. we have put this in the flaky tests using ~~helper~~ helpers attribute retryAnalyzer

12. Does your framework take Screenshot on test failure? How did you implement it? (try catch not recommended)

A. TestNG implements whenever test fails it comes to Listener on TestFailure method here where we are implementing screenshot code

(13) Selenium framework Architecture Diagram

(5)



Question 28

Max. score: 50.00

Range game

You are at the range. You have N targets labeled 1 to N from left to right.

Hit the target by performing the following operation:

- In each round starting from the first element and moving from left to right, hit every alternate target.
- Repeat the above step until only one target is left.

Note: You will get i points when you hit a target labeled i .

Calculate the score you will obtain by continuously hitting targets until there is only one target remaining.

Function description

Complete the *Solve()* function. This function takes the following argument and returns the total number of points you will get by hitting targets until only one target is left:

- N : Represents the total number of targets

Input format

- The first line contains an integer N denoting the total number of targets.

Output format

Print the total points you will get by hitting the targets by following the specified operation.

16

Constraints

17

$$1 \leq N \leq 10^6$$

18

Sample input



Sample output



19

5

11

20

21

22

23

Explanation

24

Given

25

- $N = 5$

26

Approach

27

targets = [1, 2, 3, 4, 5]



28

Initial Points = 0

29

1. Targets = [1, 2, 3, 4, 5]. Targets to be removed are [1, 3, 5]. Points = $0+1+3+5=9$.
2. Targets = [2, 4]. Targets to be removed are [2]. Points = $9+2=11$.

30

targets = [4], only one target is left therefore return total points i.e. 11.

30

< Previous Question

Next Question >

Max. score: 20.00

The contest

In a contest with N participants, find the score of the participant who ranked fourth among all the participants in terms of their score.

Function description

Complete the `solve()` function. The function takes 2 parameters and returns the 4^{th} highest score:

- N : Represents the length of the array denoting the number of participants
- arr : Represents the array of scores of participants

Input format for custom testing

Note: Use this input format if you are testing against custom input or writing code in a language where we don't provide boilerplate code.

- The first line contains N denoting the length of the array.
- The second line has an array arr containing the numbers.

Output format

Print a single integer representing the 4th highest score amongst all the participants.

Constraints

$$4 \leq N \leq 10^6$$

16

Sample input

5
9 7 5 3 1

17

18

19

20

21

Explanation

22

The numbers in decreasing order - [9, 7, 5, 3, 1]. Hence, the person with score 3 wins the mega prize.

23

Note:

24

Your code must be able to print the sample output from the provided sample input. However, your code is run against multiple hidden test cases. Therefore, your code must pass these hidden test cases to solve the problem statement.

25

Limits

26

Time Limit: 1.0 sec(s) for each input file

Memory Limit: 256 MB

Source Limit: 1024 KB

27

Scoring

28

Score is assigned if any testcase passes

29

Allowed Languages

Bash, C, C++14, C++17, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java 8, Java 14, Java 17, JavaScript(Node.js), Julia, Kotlin, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, Python 3.8, Racket, Ruby, Rust, Scala, Swift, TypeScript, Visual Basic

30

[← Previous Question](#)[Next Question →](#)

Max. score: 20.00

Question 30**Check the anagram**

Given lowercase string inputs s1, s2, and s3. Check if these strings are anagrams of each other or not. For example, strings "dog" and "god" are anagrams, so are strings "baba" and "aabb", but strings "abac" and "caaba" are not.

Function description

Complete the *solve()* function. This function takes the following 3 parameters and returns either YES or NO:

- s1: Represents input string 1
- s2: Represents input string 2
- s3: Represents input string 3

Input format for custom testing

Note: Use this input format if you are testing against custom input or writing code in a language where we don't provide boilerplate code.

- The first line contains a string s1.
- The second line contains a string s2.
- The third line contains a string s3.

Output format

Print YES if all the given 3 strings are anagrams of each other, else print NO.

16

 $1 \leq |s1|, |s2|, |s3| \leq 10^5$

17

Sample input



Sample output



18

```
bbac  
abcb  
acbb
```

YES

19

20

21

22

Explanation

23

The explanation is given in the problem statement.

24

Note:

25

Your code must be able to print the sample output from the provided sample input. However, your code is run against multiple hidden test cases. Therefore, your code must pass these hidden test cases to solve the problem statement.

26

Time Limit: 1.0 sec(s) for each input file

Memory Limit: 256 MB

Source Limit: 1024 KB

27

Scoring

28

Score is assigned if any testcase passes

29

Allowed Languages

Bash, C, C++14, C++17, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java 8, Java 14, Java 17, JavaScript(Node.js), Julia, Kotlin, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, Python 3.9, Racket, Ruby

30

[Previous Question](#)

[Next Question](#)

Here's the roadmap I shared with them:

Week 1: Core Java (or Python)

OOP Principles (Inheritance, Polymorphism, Encapsulation, Abstraction)

Exception Handling

Collections Framework (List, Set, Map)

String, Array, File Handling

Multithreading Basics

Lambda Expressions & Streams

Week 2: SELENIUM

Setting up WebDriver

Locators (XPath, CSS Selectors, ID, etc.)

Handling Web Elements (Buttons, Dropdowns, Frames, Alerts)

Synchronization (Explicit & Implicit Waits)

Page Object Model (POM)

TestNG/JUnit for Test Automation Framework

Week 3: API Testing

REST Basics (GET, POST, PUT, DELETE)

HTTP Methods and Status Codes

Tools like Postman or Swagger

Writing API Tests with REST Assured (Java) or Pytest (Python)

Validating JSON Responses

Week 4: CI/CD Pipelines

Version Control with Git

Introduction to Jenkins/TeamCity

Setting Up Test Pipelines

Triggering Tests Automatically

Integrating Tests with Build Pipelines

Week 5: Mobile Testing

Setup with Appium

Emulator and Simulator setup

Weeks 6-8: ADVANCED

Building Modular Automation Frameworks

Integrating with Reporting Tools (Allure/Extent Reports)

Parallel Execution (TestNG or JUnit)

Cross-Browser Testing

Advanced Concepts like Software Architectures, Generative AI for Testing



Home



Video



My Network



Notifications



Jobs



All

Articles

...more topics

Master these Topics to crack your next Test Automation Interview

Most companies, even before conducting technical interviews focuses on Coding rounds.

Here are some common logic programs that you might encounter in a QA Automation Interview (If you code using Java)

Core Java Concepts:

└ String Manipulation:

- * Reversing a string
- * Palindromes
- * Counting vowels or consonants
- * Finding the longest/shortest word

└ Array Manipulation:

- * Finding the largest/smallest element
- * Sorting arrays (bubble sort, insertion sort, selection sort, merge sort, quick sort)
- * Reversing an array
- * Finding duplicates
- * Finding longest substring
- * Finding repetitive substrings
- * Finding second smallest/largest element

└ Number Operations:

- * Prime number check
- * Fibonacci series
- * Factorial calculation
- * Armstrong number check

└ Data Structures:

* Implementing basic data structures like linked lists, stacks, queues

- * Traversing and searching in these structures

└ OOP Concepts:

- * Inheritance, polymorphism, encapsulation, abstraction
- * Creating abstract classes and interfaces

└ Pro-Tip:

➡ Practice Regularly

➡ Solve coding challenges on platforms like LeetCode, HackerRank

➡ Understand the Underlying Concepts



Home



Video



My Network



Notifications



Jobs



Saved Posts and Articles

All

Articles

- ◆ What is the final keyword in Java?
- ◆ What is a constructor? Can we override a constructor?
- ◆ Where is a constructor used in your framework?
- ◆ What is abstraction?
- ◆ How to achieve multiple abstraction in Java?
- ◆ What is an interface?
- ◆ What is encapsulation? Where have you used it in your project?
- ◆ What is polymorphism? Where have you applied it in your framework?
- ◆ What is method overloading and method overriding? Explain both and where you used them in your framework?
- ◆ Have you used collections in your framework?
- ◆ What is the static keyword in Java?
- ◆ What is memory management in Java?
- ◆ What is garbage collection in Java?
- ◆ Difference between constructor overloading and method overloading?
- ◆ What are exceptions in Java?
- ◆ List some exceptions you faced during automation and how you handled them.

Selenium & Web Automation

- ◆ How do you handle dynamic web elements?
 - ◆ Write the syntax of the Select class and explain its methods.
 - ◆ How do you handle iFrames?
 - ◆ How do you determine whether an element is an iFrame or not?
 - ◆ How do you handle mouse actions in Selenium?
 - ◆ How do you handle keyboard actions in Selenium?
 - ◆ How do you press the ENTER key using the Actions class?
- Write the syntax.
- ◆ How do you use the SHIFT key to type uppercase letters in Selenium? Write the syntax.
 - ◆ How to handle dynamic elements?
 - ◆ How to print all the links from a webpage? Where do you store those links?
 - ◆ How to check for broken links? How would you validate each link if there are 20 links on a page?
 - ◆ What is the finally block in Java?
 - ◆ Difference between throw and throws?
 - ◆ Difference between findElement() and findElements()



Home



Video



My Network



Notifications



Jobs



characters and concatenate each character using the `StringBuilder`'s `append()` method.

Fourth highest number in an array program

To find the fourth highest number in an array, you can sort the array and then access the element at the correct index. Alternatively, you can find the fourth highest number using a more efficient approach, like iterating through the array and keeping track of the top four distinct numbers.

Here's how you can implement both approaches:

1. Using Sorting:

```
import java.util.Arrays;

public class FourthHighest {
    public static void main(String[] args) {
        int[] arr = {10, 4, 5, 90, 120, 40, 25};

        // Sort the array in ascending order
        Arrays.sort(arr);

        // Check if the array has at least 4 elements
        if (arr.length < 4) {
            System.out.println("Array has less than
} else {
    // Access ↓ fourth highest element (a
    int fourthHighest = arr[arr.length - 4]
    System.out.println("The fourth highest
```

Message ChatGPT



Search



Reason





For the input array {10, 4, 5, 90, 120, 40, 25}:

- First highest: 120
- Second highest: 90
- Third highest: 40
- Fourth highest: 25

The output would be:

The fourth highest number is: 25

write a program to print n terms of the following series and also print sum and average of the series
33,34,36,39

Here's a Java program that prints the first n terms of the series 33, 34, 36, 39, ..., and also calculates and prints the sum and average of the series.

Explanation of the Series:

- The series starts at 33.
- The differences between consecutive terms are 1, 2, 3, and then the pattern repeats. So

Message ChatGPT



Search



Reason





Enter the number of terms (n): 10

The series is:

33 34 36 39 40 41 43 46 47 48

Sum of the series: 407

Average of the series: 40.7

Series Pattern:

- For n = 10, the first 10 terms of the series are: 33, 34, 36, 39, 40, 41, 43, 46, 47, 48.
- The sum of these terms is 407, and the average is 40.7.

write a method that will return nth largest number from a data of M numbers in the most efficient way using selection sort

To find the nth largest number from a data set of M numbers using **Selection Sort**, you can implement a variation of the algorithm where you don't have to fully sort the array. Instead, you can run the selection process for n iterations to find the nth largest element, which improves efficiency over fully sorting the array.

Here's a method that does ↓ hat:

Message ChatGPT



Search



Reason





Time Complexity:

- **Best Case Time Complexity:** $O(n * M)$, where n is the number of iterations to find the n th largest number and M is the total number of elements. Since n is typically much smaller than M , this is more efficient than fully sorting the array (which would take $O(M^2)$ in the case of full selection sort).

write $n*n$ pattern as given below if $n = 5$, where $1 \leq n \leq 12$

Here's a Java program that prints an $n * n$ pattern as shown below, where $n = 5$. The program will work for values of n where $1 \leq n \leq 12$.

Pattern for $n = 5$:

```
1  2  3  4  5  
10 9  8  7  6  
11 12 13 14 15  
20 19 18 17 16  
21 22 23 24 25
```

Program to Print the Pattern:

```
import java.util.Scann ↓  
public class NumberPattern {
```

Message ChatGPT



Search



Reason





tha

You're welcome! If you try this out, it should give you the $n * n$ pattern you're looking for. If you have any other Java-related questions or need more help with patterns, feel free to ask. Happy coding! :)

the pattern has be like this

10
12 13
15 16 17
19 20 21 22
24 25 26 27 28

Ah, I see! You want a pattern where the numbers increase row by row, starting with a base value, and the number of elements in each row increases as well.

For $n = 5$, the pattern looks like this:

10
12 13
15 16 17
19 20 21 22
24 25 26 27 28



In this case, the pattern starts at 10 and each row

Message ChatGPT



Search



Reason





Example Output for n = 3:

```
A B C  
D E F  
G H I
```

This pattern will work for any value of n between 1 and 26, and it will wrap around the alphabet if necessary!

the pattern is
11 12 13 14 15
12 13 14 15 11
13 14 15 11 12
14 15 11 12 13

Got it! You want a pattern where the numbers increase starting from 11, but they "wrap around" once they reach 15. For n = 5, the pattern looks like this:

```
11 12 13 14 15  
12 13 14 15 11  
13 14 15 11 12  
14 15 11 12 13  
15 11 12 13 14
```

Here's a Java program that generates this pattern:



Java Program:

Message ChatGPT



Search



Reason





```
13 14 15 11 12  
14 15 11 12 13  
15 11 12 13 14
```

This program will work for any value of n within the specified range ($1 \leq n \leq 12$). Let me know if you have any questions!

How rectangle is related to square in terms of inheritance? create java classes for rectangle and square with proper constructors, getters and setters methods. also write methods to find area of rectangle and square.

In object-oriented programming, **inheritance** allows one class to inherit the properties and behavior of another. A **square** is a special case of a **rectangle**, where all sides are equal. Thus, in terms of inheritance, the Square class can inherit from the Rectangle class, because a square "is-a" rectangle, but with a constraint that both the length and width are equal.

Here's how we can model this relationship in Java:

- **Rectangle class:** It has two properties: length and width, and methods to calculate the area of a rectangle.

Message ChatGPT



Search



Reason

