Self-Introduction:

Hi My name is "<YourName>"

I have total "<NoOfYears>" years of experience in both Manual and automation testing.

In Which I have "<NoOfYears>" years of experience in UI automation using Selenium - Java

And

"<NoOfYears>" year of experience in API automation testing using RestAssured and Postman

In Frameworks I have experience in Cucumber, TestNG, Data Driven and Keyword Driven.

And also we used various components in our frameworks like Maven build, GIT Version Control System, Jenkins and Page Object Model.

Coming to my roles and responsibilities

In manual testing, I have involved in Analyzing the requirements and user stories to Test Closure phase.

I involved in identifying the scenarios, creating the test cases, execution of test cases and reporting the defects. Also involved in performing smoke, sanity and manual regression testing.

Coming to the automation,

I have developed test scripts for UI manual cases using selenium Java after defining the automation steps.

And API automation side, I have developed automation scripts using RestAssured Java library.

It is also a cucumber framework with TestNG.

Coming to the SDLC process, We followed Agile methodology, it's 2 weeks Sprint based process.

I have involved in all the ceremonies of Agile.

Manual Testing FAQ's:

1. What are all the Agile Ceremonies/Meetings you have attended?

Ans: I have attended ceremonies like Backlog grooming and refinement session,
Sprint planning -> Based capacity, we take the user stories, Daily Scrum calls and status calls, Sprint Demo, Sprint Retrospective meeting -> In last sprint what went well, what went wrong, What improvements required for next sprint.

2. What is Sprint Retrospective.

Ans: In Sprint retrospective meeting, we discuss What went well in the sprint, what went wrong in the sprint, what are all improvements required for the next sprint.

3. How do you estimate (Provide story points) User Stories.

Ans: First we check the scenarios needs to be covered as part of the story, what are the dependencies and no modules involved in that story, based on that we provide story pints.
We following Fibonacci series -> 1, 2, 3, 5, 8

4. What all are the Testing Techniques you followed while testing the User Story.

Ans: We used various testing techniques based on the requirement to reduce the no of test cases. I followed Equivalence Class Partition technique for the input data and Boundary Value Analysis to check boundary conditions and edge cases and State Transmission technique and also decision table techniques while preparing the test cases.

5. What is difference between Smoke, Sanity and Regression Tests.

Ans: Smoke Test: In smoke Test we navigate to each and every page and we verify whether the page content is properly loading or not and all the options are available or not and also if we have any important links or important popups those are working or not we check. This test we also called it as Build Verification Test.

Sanity Test: In Sanity Test we have some important functionalities, those we validate by providing input data and also we check expected results. We follow test case document and we check expected results.

Regression Test: In Regression Test we validate all functionalities to check whether the newly deployed stories are breaking the existing functionality. And also to check side effects of new stories. We perform this test in the Staging/Preprod environment.

6. **What is defect life cycle.**

Ans: When we raise the defect, it will be 'New' state, once it is assigned to developer, it will be in 'Assigned' state, once it is assigned to the developer, it will 'Open' state, If developer accepts the defects he will work on the defect, he changes to 'Fixed' state. Once it is fixed we will retest the defect and We will 'Close' the defect. If the defect is replicating, we change the status to 'ReOpen'.

If Developer don't accept the defect, he can 'Reject' the defect, If it is a duplicate defect, he change the status to 'Duplicate', If it is not fixable this time, he changes to 'Deferred' state with proper comments.

7. What is the difference between severity and priority.

Ans: **Severity:** It is degree impact, how much the defect is affecting the application based on that we assign the severity. We have Critical, High, Medium and Low severities we assign

Priority: If we multiple defects, among all defects, which one needs to be fixed first. That we decide in the defect priority/Defect Triage call. We assign the Priorities like High, Medium and Low.

8. Explain Agile Process.

Ans: In agile we take the requirement and we divide into small user stories.
These user stories we target to complete in sprint wise, In each sprint we have 2 weeks of time.

For each and every story we perform Analysis, Design, Coding, Testing. It is a iterative process.

9. How do you handle the situation when you receive the user story for execution at the last day of the sprint.

Ans: I try to extend my timings and try to complete that story. If it taking more time, initially I identify that risk and I will inform to Scrum master in the dial scrum call like it might spill over in QA Testing.

10. What is Test Plan and What it contains?

Ans: Test Plan is a document which contains Test Strategy, Test Estimations, Resources Details, Risks, Testing scopes and Schedules.

11. What is the difference between Test Plan and Test Strategy.

Ans: Test Plan contains Test Strategy, Test Estimations, Resources Details, Risks, Testing scope and Schedules.

Coming to Test Strategy, we will discuss, how we can simply test that user stories and what are the technique's we need to follow to test user stories.

Automation Testing FAQ's:

Java Interview Question:

1. Explain OOPs concepts.

Ans: We have Object Oriented concepts like Encapsulation, Inheritance, Polymorphism and Abstraction.

Encapsulation means we bind the data with the variables and we perform the operations on data with the help of methods and we provide security to our data with access modifiers and provide the access with access modifiers and we can initialize the data with constructor. Whenever we required we can create a Object of class and we can access the methods. In our Page Object classes I have implemented encapsulation by defining the WebElemets with private access modifier and also created the methods with public access modifiers whenever we required I created a object of that class and accessed the class methods.

Inheritance means access the one class properties/members directly into another class by providing the relation between the classes using extends keyword. After providing the relation we can create a object for child class and we can access both parent and child class members. In our project we have base class which has common methods and also to initialize the driver we have constructor, that we have extended to our Page Object classes.

Polymorphism means we can use the same method name for different logics, it is two types, one is Overloading and another one is overriding.

Overloading means we can write the methods with same name but the parameters count or type must different from method to method. In our Page classes whenever I want create multiple methods with same name I have created but I have passed some time only locator, only WebElement, only By type.

Overriding means, whenever we are not satisfying the existing logic of parent class method we can create method with same name and same parameters with different login. In our BaseClass we have common methods, if I'm unable to use common method as per my requirement I have created a new method in my ChildClass with same name and same parameters and I have override it.

Abstraction means hiding the actual logic and providing the overview of logic with abstract methods. we can perform abstraction with Abstract classes and Interfaces. We didn't create any Abstract class or interface but I have used existing interfaces like WebDriver, JavaSriptExecutor, TakeScreenshot, WebElement, Alert.

2. What is static keyword.

Ans: Static allocates memory by itself it will be recognized by JVM and it allocates memory during class loading time. We can use static keyword for variables, methods and nested classes.

3. Can we override static methods.

Ans: We cannot override static methods because it maintains separate static memory for each class

4. what is difference between final, finally and finalize keywords.

Ans: Final is a keyword we can use it for variables, methods, classes. If we use final keyword for variables, we cannot change variable value, if use final keyword for methods, we cannot override and if we use final keyword for classes we cannot extend.

Finally is a block, we use it along with try and catch blocks, whether exception occur or don't occur, it always executes. Most of the time we use it for database connections.

Finalize() is method it is automatically called by JVM to clear the garbage.

5. what is difference between Compile-time Polymorphism and Runtime-Polymorphism.

Ans: Method overloading is a compile-time polymorphism and Method overriding is run-time polymorphism.

6. What is difference between Abstract Class and Interface.

Ans: We can use Abstract class for both abstract methods and non-abstract methods.

But in Interface we can write only Abstract methods. In both the cases we cannot create the Object, We have to extend or implement the Abstract class and Interface into child classes but in Abstract we can also create Constructor, that we can call into child class.

7. What is difference between StringBuilder and StringBuffer.

Ans: StringBuilder and StringBuffer to make the string mutable.
StringBuilder cannot support for Multi-threading because method are not Synchronized.

StringBuffer supports multi-threading because methods are thread-safe.

8. How do you handle the exceptions?

Ans: We have unchecked exceptions and checked exceptions.
Unchecked exceptions we can ignore using 'throws' keyword or we can also use try catch blocks.

Whereas Runtime time exceptions we have handle with try and catch blocks.

9. What is difference between Checked Exceptions and Un-checked exception?

Ans: Checked exceptions are compile time exceptions we have handle before executing the logic. We have checked exception like FileNotFoundException, IOException, SQLException, IntruptedException. We can use throws keyword also to ignore it.

Un-checked exceptions are run-time exceptions we have to handle with try-catch block, these exceptions occur with the data. Exception like ArithmaticException, NullPointerException, ArrayIndexOutOfBoundsException, StringIndexOutOfBoundsException, ClassCastException, NumberFormatException etc.

10. What is the use of 'this' and 'super' keywords

Ans: 'this' keyword represents current object, we can access parent and child instance members with this keyword.

'super' keyword represents parent class object we can access parent class members with super keyword.

11. What is functional Interface.

Ans: An Interface which contains only one abstract called functional interface. We can use lambda expression for the functional interfaces. We have functional interfaces like Function, BiFunction, Predicate, Consumer, Supplier Comparator interfaces

12. What are the collections you used in your automation frameworks?

Ans: I have user List, Set, Map

Under List I have used ArrayList and LinkedList.

Under Set I have used HashSet, LinkedHashSet and TreeSet.

Under Map I have used HashMap, LinkedHashMap, TreeMap and HashTable.

13. What is difference between List and Set.

Ans: List accepts duplicate values and also it preserves the order of the elements while adding. And also we have get() method to access List elements by using index.

Set doesn't allow duplicates and it won't preserve the order while adding the elements and we don't have any get() method to access the Set element, we have to use enhanced for loop or iterators.

14. What is difference between HashMap and HashTable.

Ans: HashMap allows one null key and multiple null values and also it won't support for multi-threading process.

HashTable doesn't allow either null key or null values and it supports multi-threading.

15. What is difference between ArrayList and LinkedList.

Ans: ArrayList follows index-based algorithm and it works very faster when accessing the elements.

LinkedList follows Head and tail algorithm, it saves the memory but compare to ArrayList it works slower.

16. Why string is immutable?

Ans: When we create ay string, it stored into string pool memory, it is constant memory. Once we create in string pool, that value cannot be changed. If modify the string it will creates a new string.

Selenium Interview Questions:

1. What are different Components of Selenium?

Ans: We have different components in Selenium like

Selenium IDE, it is browser add-in, it specific to each and every browser. Data driven is not possible with Selenium IDE

Selenium RC -> it dependent on Remote Control server, we have to start Remote Server to use Selenium RC.

Selenium WebDriver -> We use driver file to interact with the browser, it will convert our selenium code into Browser understandable language using JSON Protocol. In selenium4, new API which follows W3C API, it can directly interact with browser without any driver file.

Selenium GRID is configuration tool, using which we can create multiple Virtual Machines and we and we can use these Virtual machines to perform Cross browser/Platform testing.

2. What is the difference between ImplicitWait and Explicit wait.

Ans: If we configure ImplicitlyWait, it will applicable for all the elements which we are using in our script, we configure it at driver level.

Explicit wait is specific for particular element which is taking more time than implicit wait.

We have two types of Explicit waits

1. WebDriverWait
2. FluentWait

3. What are the selenium exceptions you faced.

Ans: I have faced Selenium Exceptions like, most of the time NoSuchElementException, ClickInterceptedException, StaleElementException and TimeOutException and ElementNotFoundException

4. How do you handle Selenium Exceptions.

Ans: We can handle Selenium exceptions by using WebDriverWait and ExpectedConditions class. We have methods VisibilityOf, StalenessOf, elementToBeClickble methods in ExpectedConditions class. With the help of those we can handle Selenium Exceptions.

5. What is difference between WebDriverWait and FluentWait.

Ans: WebDriverWait we use to handle Selenium Exceptions and we have separate methods for each Exception and WebDriverWait retry for the same element for each and every 500 milli seconds in catch block. In FluentWait that retry polling time we can increase by using pollingEvery() method and we can ignore any Exception using ignoring() method.

6. What is difference between WebDriver and ChromeDriver.

Ans: WebDriver is an Interface and ChromeDriver is a class, WebDriver is Parent Interface of all the browser classes in Selenium, we can create object of child class and we can assign it to Parent Interface.

7. How do you handle the Alerts.

Ans: To handle the Alerts we have to switch to the Alert by using switchTo() and alert() methods. Then we can accept() or dismiss() the alerts.

8. How do you handle Multiple Windows and Tabs.

Ans: If we want to handle the multiple Windows or Tabs, we have to take the Window id's by using getWindowHandle() and getWindowHandles() methods.

getWindowHandle() return window Id as string, that we will store into one string variable.

getWindowHandles() returns set of strings (Set<String>) of all window Id's, we can read one by one window Id and that we can pass it to switchTo().window() method and we can switch to the window. If we want to come back to the parent window, we have switch with parent window id.

9. how do you handle the frames.

Ans: To handle the frames we have to identify the frame by using Id, name or any WebElement

then we can use id/name/WebElement in SwitchTo().frame() method and we can switch to frame, we can perform the action in the frame. If we want to switch back to the main HTML page, we can use switchTo().defaultContent() method.

10. How do you handle StaleElementException?

Ans: We handle StaleElementException by using WebDriverWait ExpectedConditions class stalenessOf() method and we can pass that element to that method.

11. How to take the screenshot.

Ans: We have TakesScreenshot interface, it has getScreenshotAs() method. We have to pass OutputType.FILE also in that method, we will get file. We need to downcast WebDriver reference that we can assign to TakesScreenshot reference.

TakesScreenshot ts = (TakesScreenshot)driver; // down-casting

12. How do you scroll the browser.

Ans: To scroll the browser, we have to use JavaScriptExecutor interface executeScript() method, we can pass JavaScript 'window.scroll(x,y)' x and coordinates we have to pass.

We have to downcast WebDriver reference and we can assign it to JavaScriptExecutor reference by down-casting it.

JavaScriptExecutor js = (JavaScriptExecutor)driver; //down-casting

13. How do you drag and drop the element.

Ans: Using Selenium Actions class we can drag and drop the element. We can use direct dragAndDrop() method and we need to pass source and destination element in that method and build() and perform() methods we have to use. And we can also use clickAndHold() and release() methods also to drag and drop the element.

Actions actions = new Actions(driver);

actions.moveToElement(sourceELement).dragAndDrop(sourceELement, destElement).build.perform(); (or)
actions.moveToElement(sourceELement).clickAndHOld(sourceELement).release(destElement).build.perform();

14. What is the use of JavaScriptExecutor?.

Ans: With the help of Java script executor we can execute JavaScript in Java code, we can pass JavaScript to executeScript method, it will be executed. We can perform scroll, click, scroll to particular element, set the attribute value also with JavaScript.

15. What is RemoteWebDriver Class.

Ans: RemoteWebDriver is fully implemented class, it is between WebDriver Interface and our bowser class. When we call WebDriver methods, those will be called from RemoteWebDriver.

16. What is TestNG.

Ans: TestNG is a unit test library, it is providing various annotations to perform common operations while we are writing the tests. With the help TestNG we can generate the reports, we can perform parallel testing, we can perform grouping, we can run the test multiple times etc..

17. What are all the TestNG annotations you have used in TestNG.

Ans: @BeforeSuit, @BeforeTest, @BeforeClass, @BeforeMethod,

@Test, @Parameters, @DataProviders

@AfterMethod, @AfterClass, @AfterTest, @AfterSuit.

18. How do you run the test cases in parallel.

Ans: In Testng.xml, in <test> tag, we have an attribute, parallel, that we have to make it as methods and also we need to increase the thread Count. To perform parallel testing, we need to initialize our Brower with ThreadLocal class.

19. What is priority in TestNG.

Ans: We use priority keyword in @Test annotations to prioritize the execution of methods. By default all the Test cases runs based on method names in alphabetical order, if we want to order the execution we can set the priority. Default priority for all test method is 0 and if we have same priority for more than one method, again it uses alphabetical order.

20. How do you pass the Test Data to our test method.

Ans: We use @DataProviders annotation that data provider we will use into our @Test method.

We have to pass the Test data in the form of Object Array, 2D-Object Array or Iterator. We can read the data from excel with the help of Apache Poi library, that logic we can use it in DataProviders method.

21. How do you run a particular test cases multiple-time.

Ans: We can use 'invocationCount' property in the @Test method and we can provide the count to run a test case multiple-times.

22: How do you upload the file?

Ans: If we are able to set the path in the file filed, we can use sendkeys() method and we can set the file path, we can also set the path using JavaScript to upload file.

If the above methods are not working, we can use Java Robot class and we perform the keyboard actions by using keyPress() and keyRelease() methods and to copy the file path to clip board, we have to use String Selection and Toolkit classes.

API Interview Questions:

1. What all are the HTTPS methods you have used.

Ans: I have tested REST Http methods like POST, PUT, PATCH, GET, DELETE.

2. How do you validate the response.

Ans: First we check the request success or not based on the status codes. Then if we have any mandatory files to validate, those we will check in the response.

3. What are success codes.

Ans: We have success codes like 200 for the GET call, 201 for POST call, 204 for the DELETE call.

4. What are client-side error codes.

Ans: We have client-side errors status codes like 400 for bad request, 401 for Un-authorized when we provide wrong credentials / wrong token, 403 for forbidden when we provide wrong client credentials, and 404 for File not found.

5. what are server-side error codes.

Ans: We have server-side error codes are 500 internal server error, 502 Bad Gateway, 503 server not found error.

6. What is difference between Authentication and Authorization.

Ans: If we want access any API which requires authentication with user name and password we use basic authentication with valid credentials and we generate Basic Token that token we use it as Authorization header. Which we call it is Authentication, Coming to authorization, few APIs requires extra Authorization to identify the user. We provide client details like Client Secret, Client name, Scope, token generation URL and we generate Bear token, that we pass it as Authorization header. Generally we use it for OAuth2 Authorization.

7. What is Static import

Ans: We can use static import to import the static content of any class and we can call that data without class name after importing.

import static package.ClassName.*;

8. What is RequestSpecification and ResponseSpecification

Ans: If we want to use common Request Configuration for all the endpoints, we can use RequestSpecification class that we can use it to all endpoints. Similarly we want to validate response with some configuration like response time, response headers and common status code, these we can configure with ResponseSpecification call and we can use it for all endpoints

9. How do you validate Response Schema.

Ans: We can validate response schema in postman using script assertion and we need to write JavaScript to validate it. In RestAssured we can create a model schema JSON file and we need pass it to matchesJsonSchemaInClasspath() method.

10. How do you validate the API response in RestAssured.

Ans: Once we get the response, from the response Object, We get the status code by using getStatusCode() method and we put Assertion to validate it.

Assert.assertEquals(200, response.getStatusCode());

To validate the fields, we can use response.jsonPath().get("filedName").toString();

Assert.assertNotNull(response.josnPath().get("filedName").toString());

Assert.assertEquals("ExpectedString, "response.jsonPath().get("filedName").ToString());;

And also we use POJO classes, And response we will de-serialize with POJO class with the help of ObjectMapper class. With the help of POJO class getter methods we read the filed values from response. Again we will use Assertions.

SQL Interview Questions:

1. Do you have any experience in Database Testing?

Ans: I didn't involve in Database Testing but I have used SQL select queries with JOINS and Aggregate functions. We used to validate from both UI and DB level. Cross Comparison we used to do. I can write select queries only. I never used Insert, Update and delete queries.

2. What type of database you used in your project?

Ans: have used Oracle database and SQL Developer tool i have used to write the SQL queries. And also I have used in IntelliJ

3. What DML commands you used ?

Ans: We have DML commands like select, insert, update, delete. But I used only select command most of the time.

4. What are the joins you used to join the tables?

Ans: I have used joins like normal Join, Inner Join, Left Join, and also Right Join.

Normalization: Creating various tables based on the category.

De-Normalization: Combining the various tables based Primary and foreign key, these keys should not be null.

Joins:

To join the tables, we should have common column in the both the tables

JOIN -> is used to combine more than one tables based on matching columns.

```
select * from emp join company on emp.empId = company.empId
```

INNER JOIN -> Returns matching records in both tables

```
select * from emp inner join company on emp.empId = company.empId
```

LEFT JOIN-> Returns all records from the left table, and the matched records from the right table.

```
select * from emp left join company on emp.empId = company.empId
```

RIGHT JOIN ->Returns all records from the right table, and the matched records from the left table

```
select * from emp right join company on emp.empId = company.empId
```

5. What are the aggregate functions.

Ans: I have used aggregate functions like

COUNT() - to get the count of all records;

```
select count(*) from emp;
```

MIN() - to get the minimum value of a particular column

```
select min(columnName) from emp;
```

MAX() - to get the maximum value of a particular column

```
select max(columnName) from emp;
```

SUM() - to get sum of all values of particular column

```
select sum(columnName) from emp;
```

AVG() - to get average value of a column

```
select avg(columnName) from emp;
```

6. How to do you sort the column values?

Ans: We can sort column by using order by keyword, we can order in ascending order by using 'asc' keyword and we can order in descending order by using 'desc' keyword

```
select * from emp order by salary asc

select * from emp order by salary desc
```

7. How do you filer particular employee based on empName or empId.

Ans: We can use where keyword and we can mention columnName = columnValue;

```
select * from emp where empId = 1001;

select * from emp where empId = '1001';

select * from emp where empName = 'Raju';
```

8. How do you get multiple records based on the condition?

Ans: We can use 'in' and within the parenthesis we can provide multiple values to filter multiple records. select * from emp where empId in (1001,1011,1022,1033,1044);

9. How do you filter emp records whose name starts with 'R.

Ans: We can use like keyword and also wild cards to filter matching records.

```
select * from emp where empname like '%R%';
```

10. How do group the columns?

Ans: We can group the column values by using 'group by' keyword

```
select * from emp group by designation.
```

11. What is Primary key?

Ans: Primary key is a unique value, we can set it to the table column, and primary key should not be duplicate and also it should not be null.

12. What is foreign key?

Ans: Foreign key is a unique value, we can set it to the table column, and foreign key should not be duplicate and also it should not be null. When we have two tables, main table contains Primary and sub table contains Foreign key.

13. How do get required number of records from the table.

Ans: We can use 'limit' keyword to limit the no of records.

```
select * from emp limit 5;
```

14. How do you use sub-queries?

Ans: If we want to use one query result as a table we can use sub-queries?

For example, If I want to filter 3rd highest salary from the emp table I can write the query like

```
select * from (select * from emp order by empSalary desc limit 5) order by empSalary
asc limit 1;
```

Coding Questions:

Java:

1. // swap two numbers variables with third variable

```
public static void swapNumbersWithThirdVariable(int a, int b) {
        System.out.println("A val: " + a);
        System.out.println("B val: " + b);
        //      swap two variables with third variable
                int c = a;
                a = b;
                b = c;
                System.out.println("A val: " + a);
                System.out.println("B val: " + b);
}
```

2. // swap two string variables without third variable

```
public static void swapNumbersWithoutThirdVariable(int a, int b) {
        System.out.println("Before swap A val: " + a);
        System.out.println("Before swap B val: " + b);
        a = a + b; // 100+200 = 300
        b = a - b; // 300-200 = 100;
        a = a - b; // 300 - 100 = 200;
        System.out.println("After swap A val: " + a);
        System.out.println("AFter swap B val: " + b);
}
```

3. // check given String/Number Palindrome or not

```
public static void swapStringsWithoutThirdVariable(String str1, String str2) {
        System.out.println("Before swap str1 val: " + str1);
        System.out.println("Before swap str2 val: " + str2);
        str1 = str1 + str2; // Hello + Java = HelloJava;
        str2 = str1.substring(0, str1.length() - str2.length()); // Hello
        str1 = str1.replace(str2, "");
        System.out.println("After swap str1 val: " + str1);
        System.out.println("After swap str2 val: " + str2);
}
```

4. // check given number Palindrome or not

```
public static void checkNumberPalindrome(int number) {
        // 121
        int revNum = 0;
        int tempNum = number;
        while (tempNum > 0) {
                int remainder = tempNum % 10; // 1, 2, 1
                revNum = (revNum * 10) + remainder; // 121
                tempNum = tempNum / 10; // 12, 1, 0
        }
        if (revNum == number)
                System.out.println(number + " is Palindrome");
        else
                System.out.println(number + " is not Palindrome");
        }
}
```

5. // check given number Palindrome or not

```
public static void checkStringPalindrome(String str) {
        String revStr = "";
        for (char eachChar : str.toCharArray()) {
                revStr = eachChar + revStr;
        }
        if (revStr.equals(str))
                System.out.println(str + " is Palindrome");
        else
                System.out.println(str + " is not Palindrome");
        }
}
```

6. // check given number Prime or not

```
public static void checkNumberPrime(int num) {
        int count = 0;
        for (int i = 1; i <= num; i++) {
                if (num % i == 0)
                count++;
        }
        if (count == 2)
                System.out.println(num + " is Prime");
        else
                System.out.println(num + " is not Prime");
        }
}
```

7. // check given number Armstrong number

```
public static void checkNumberArmstrong(int number) {
	// 153
	int armNum = 0;
	int tempNum = number;
	while (tempNum > 0) {
		int remainder = tempNum % 10; // 3, 5, 1
		armNum = armNum + (remainder * remainder * remainder); // 27, 152, 153
		tempNum = tempNum / 10; // 15, 1, 0
	}
	if (armNum == number)
		System.out.println(number + " is Armstrong number");
	else
		System.out.println(number + " is not Armstrong number");
	}
}
```

8. // Print Fibonacci series

```
public static void fibonacciSeries(int num) {
	int a = 0;
	int b = 1;
	System.out.print(a + " ");
	for (int i = 0; i <= num; i++) {
		System.out.print(b + " ");
		int c = b;
		b = a + b;
		a = c;
	}
}
```

9. // Print Right Half Pyraid

```
public static void printRightHalfPyramid(int num) {
	for (int i = 1; i <= num; i++) {
		for (int j = 1; j <= i; j++) {
			System.out.print("* ");
		}
		System.out.println("");
	}
}
```

10. // Print Right Half Pyraid

```
public static void printReverseRightHalfPyramid(int num) {
	for (int i = num; i > 0; i--) {
		for (int j = 1; j <= i; j++) {
			System.out.print("* ");
		}
		System.out.println("");
	}
}
```

11. // Print Given Highest Number

```
	public static void printGivenHighestNumberFromArray(int[] intArr, int givenNumber) {
		if (intArr.length - 1 >= givenNumber) {
			for (int i = 0; i < intArr.length; i++) {
				for (int j = i + 1; j < intArr.length; j++) {
					if (intArr[i] > intArr[j]) {
						int temp = intArr[i];
						intArr[i] = intArr[j];
						intArr[j] = temp;
					}
				}
			}
			System.out.println(givenNumber + "st/nd/rd highest number: " +
intArr[intArr.length - givenNumber]);
		} else {
			System.out.println(givenNumber + " index not found");
		}
	}
```

12. // Find repeating numbers from Array

```
public static void findRepeatingNumbersFromArray(int[] intArr) {
        for (int eachNumber : intArr) {
                int count = 0;
                for (int eachNumber1 : intArr) {
                        if (eachNumber == eachNumber1)
                                count++;
                        }
                        if (count >= 2)
                                System.out.println(eachNumber + " is repeating number");
                                count = 0;
                }
        }
```

13. // Find repeating chracters from a string

```
public static void findRepeatingCharactersFromString(String str) {
        for(char eachChar : str.toCharArray()) {
                if( str.indexOf(String.valueOf(eachChar)) !=
str.lastIndexOf(String.valueOf(eachChar))) {
                        System.out.println(eachChar+" is a repeating character");
                }
                        str = str.replaceAll(String.valueOf(eachChar),"");
        }
}
```

14. // Reverse Words In A Sentence Without Changing Their Position

```
public static void reverseWordsInASentenseWithoutChangingTheirPostion(String sentense) {
	String revSrentense = "";
	for(String eachWord : sentense.split(" ")) {
		String revWrod = "";
		for(char eachChar : eachWord.toCharArray())
			revWrod = eachChar + revWrod;
			revSrentense = revSrentense + revWrod + " ";
		}
		System.out.println(revSrentense.trim());
	}
}
```

15. // Find Repeating Words and its Count From sentence

```
public static void findRepeatingWordsAndItsCountFromSentense(String sentense) {
	for(String eachWord : sentense.split(" ")) {
		int count = 0;
		for(String eachWord2 : sentense.split(" ")) {
			if(eachWord.trim().equals(eachWord2.trim()))
				count++;
			}
			if(count > 1)
				System.out.println(eachWord + " is repeating word and
repeated "+ count+ " times");
			sentense = sentense.replaceAll(eachWord, "").trim();
		}
	}
}
```

16. Find Vowel Characters From a String

```
public static void findVowelCharactersFromString(String str) {
	for(char eachChar : str.toCharArray()) {
		if("aeiou".contains(String.valueOf(eachChar).toLowerCase()))
			System.out.print(eachChar);
		}
	}
}
```

Selenium:

1. Write a Program to Handle Multiple Windows/Tabs.

Ans:

```
String mainWindowId = driver.getWindowHandle();
Set<String> allWindowIds = driver.getWindowHandles();
System.out.println(allWindowIds);
for (String eachWindowId : allWindowIds) {
	if (!eachWindowId.equals(mainWindowId)) {
		driver.switchTo().window(eachWindowId);
		//perfrom the action
		driver.close();
		driver.switchTo().window(mainWindowId);
	}
}
```

2. Write a Program to Handle Frames.

Ans:

```
driver.switchTo().frame("frame1");
// Perform the action in Frame
driver.switchTo().defaultContent();
```

3. Write a program to Handle Alerts.

Ans:

```
Alert alert1 = driver.switchTo().alert();

alert1.accept(); (Or)

alert1.dismiss();
```

4. Write a Program to Double click the element.

Ans:

```
Actions actions = new Actions(driver);

actions.moveToElement(chooseFileElement).doubleClick(chooseFileElement).build().perform();
```

5. Write a program to drag and drop the element.

Ans:

```
Actions actions = new Actions(driver);

actions.moveToElement(chooseFileElement).contextClick(chooseFileElement).build().perform();
```

6. Write a program to right lick the element.

Ans:

```
Actions actions = new Actions(driver);

actions.moveToElement(chooseFileElement).dragAndDrop(chooseFileElement,chooseFileEleme
nt).build().perform();
```

7. Write a program to send the Keyboard events to the element.

Ans:

```
Actions actions = new Actions(driver);

actions.moveToElement(chooseFileElement).sendKeys(Keys.CONTROL+"A").build().perform();
```

8. Write a program to upload the file.

Ans:

```
StringSelection selectStr = new StringSelection(filePath);
Toolkit.getDefaultToolkit().getSystemClipboard().setContents(selectStr, null);
Robot robot = new Robot(); // we can perform mouse and keyboard actions
robot.keyPress(KeyEvent.VK_CONTROL);
robot.keyPress(KeyEvent.VK_V);
robot.keyRelease(KeyEvent.VK_CONTROL);
robot.keyRelease(KeyEvent.VK_V);
robot.keyPress(KeyEvent.VK_ENTER);
robot.keyRelease(KeyEvent.VK_ENTER);
```

9. Write a program to handle the stale element exception.

Ans:

```
WebDriverWait wait = new WebDriverWait(driver, 20);
wait.until(ExpectedConditions.stalenessOf(element));
```

10. Write a program to handle the click intercepted exception.

Ans:

```
WebDriverWait wait = new WebDriverWait(driver, 20);
wait.until(ExpectedConditions.elementToBeClickable(element));
```

11. Write a program for FluentWait.

Ans:

```
FluentWait<WebDriver> wait = new FluentWait<>(driver);
wait.withTimeout(Duration.ofSeconds(20)).pollingEvery(Duration.ofSeconds(5)).ignoring(NoSuchElementException.class);
Function<WebDriver, WebElement> element1 = new Function<WebDriver, WebElement>() {
    @Override
    public WebElement apply(WebDriver webDriver) {
        return element;
    }
};
Function<WebDriver, WebElement> element2 = (driver) -> element;
```

12. Write a program to scroll the webpage using JavaScript.

Ans:

```
JavascriptExecutor je = (JavascriptExecutor) driver; // down-casting
je.executeScript("window.scroll(0,document.body.scrollTop)");
je.executeScript("window.scroll(0,document.body.scrollLeft)");
```

13. Write a program to scroll to particular element.

Ans:

```
JavascriptExecutor je = (JavascriptExecutor) driver; // down-casting
```

14. Write a Program to click element using JavaScript.

Ans:

```
JavascriptExecutor je = (JavascriptExecutor) driver; // down-casting
```

15. Write a program to Take the screenshot.

Ans:

```
TakesScreenshot ts = (TakesScreenshot) driver;

File screenshot = ts.getScreenshotAs(OutputType.FILE);

String screenshotName = new SimpleDateFormat("yyyy-mm-dd hh:mm:ss").format(new
Date()).replaceAll("[^0-9]", "") + ".jpg";

String screenshotDirectory = System.getProperty("user.dir") +
Utils.getTestProperty("screenshots.path");

screenshotPath = screenshotDirectory + screenshotName;

FileUtils.copyFile(screenshot, new File(screenshotPath));
```

API

1. Write a program to perform the GET request and validate the response.

Ans:

```
Response response = given().when().get("https://reqres.in/api/unknown/2").thenReturn();

System.out.println(response.getStatusCode());

System.out.println(response.body().asString());

Assert.assertEquals(200, response.getStatusCode());

Assert.assertNotNull(response);

JsonPath jsonPath =  response.body().jsonPath();

String data = jsonPath.get("data").toString();
```

2. Write a program to perform the POST request and validate teh response.

Ans:

```
RestAssured.baseURI = "https://reqres.in";

String requestBody = "{\r\n" + "    \"name\": \"morpheus\",\r\n" + "    \"job\": \"leader\"\r\n" +
"}";

var validateRes = given().with().contentType(ContentType.JSON).header("Accept",
"*/*").and().body(requestBody).when().post("/api/users").then().log().ifValidationFails().body(Is
Null.notNullValue()).statusCode(201);
```

3. How do you pass the Headers in RestAssured.

Ans:

```
RestAssured.baseURI = "https://reqres.in";

String requestBody = "{\r\n" + "    \"name\": \"morpheus\",\r\n" + "    \"job\": \"leader\"\r\n" +
"}";

var validateRes = given().with().contentType(ContentType.JSON).header("Accept",
"*/*").and().body(requestBody).when().post("/api/users").then().log().ifValidationFails().body(Is
Null.notNullValue()).statusCode(201);

String filePath = System.getProperty("user.dir")  +
"\\src\\test\\resources\\TestData\\UserServiceTestData\\CreateUser.json";

ObjectMapper objMapper = new ObjectMapper();

CreateUser createUserDetailsFromFile = objMapper.readValue(Paths.get(filePath).toFile(),
CreateUser.class);

var validateRes3 = given().with().contentType(ContentType.JSON).header("Accept", "*/*").and()

.body(createUserDetailsFromFile).when().post("/api/users").then().log().ifValidationFails()

.body(IsNull.notNullValue()).statusCode(201);

Map<String, String> createUserRes3 = validateRes3.extract().as(Map.class);

System.out.println(createUserRes3);

Assert.assertEquals(createUserRes3.get("name"), createUserDetailsFromFile.getName());

Assert.assertEquals(createUserRes3.get("job"), createUserDetailsFromFile.getJob());

Assert.assertNotNull(createUserRes3.get("id"));
```

4. How do you pass the query Parameters.

Ans:

```
RestAssured.baseURI = "https://reqres.in";

Response res = given().with().queryParam("page", "2").when().get("/api/users").thenReturn();

Assert.assertEquals(res.getStatusCode(), 200);

JsonPath jsonpath = res.body().jsonPath();

List<Map<String, Object>> dataList = jsonpath.getList("data");

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

5. How do you pass the path parameters.

Ans:

```
RestAssured.baseURI = "https://reqres.in";

Response res = given().with().pathParam("id", "2").when().get("/api/users/{id}").thenReturn();

Assert.assertEquals(res.getStatusCode(), 200);

JsonPath jsonpath = res.body().jsonPath();

System.out.println(jsonpath.get("data").toString());
```

6. How do you de-serialize the response using POJO class.

Ans:

```
CreateUser createUserDetails = new CreateUser();

createUserDetails.setName("Raju");

createUserDetails.setJob("Architect");

var validateRes2 = given().with().contentType(ContentType.JSON).header("Accept", "*/*").and()
.body(createUserDetails).when().post("/api/users").then().log().ifValidationFails().body(IsNull.no
tNullValue()).statusCode(201);

Map<String, String> createUserRes2 = validateRes2.extract().as(Map.class);

System.out.println(createUserRes2);

Assert.assertEquals(createUserRes2.get("name"), createUserDetails.getName());

Assert.assertEquals(createUserRes2.get("job"), createUserDetails.getJob());

Assert.assertNotNull(createUserRes2.get("id"));
```