Ques.1. What is Selenium?

Ans. Selenium is a robust test automation suite that is used for automating web based applications. It supports multiple browsers, programming languages and platforms.

Ques.2. What are different forms of selenium?

Ans. Selenium comes in four forms-
Selenium WebDriver - Selenium WebDriver is used to automate web applications using browser's native methods.
Selenium IDE - A Firefox plugin that works on record and play back principle.
Selenium RC - Selenium Remote Control(RC) is officially deprecated by selenium and it used to work on javascript to automate the web applications.
Selenium Grid - Allows selenium tests to run in parallel across multiple machines.

Ques.3. What are some advantages of selenium?

Selenium is open source and free to use without any licensing cost.
It supports multiple languages like Java, ruby, python etc.
It supports multi browser testing.
It has good amount of resources and helping community over the internet.
Using selenium IDE component, non-programmers can also write automation scripts
Using selenium grid component, distributed testing can be carried out on remote machines possible.

Ques.4. What are some limitations of selenium?

We cannot test desktop application using selenium.
We cannot test web services using selenium.
For creating robust scripts in selenium webdriver, programming langauge knowledge is required.
We have to rely on external libraries and tools for performing tasks like - logging(log4J), testing framework-(testNG, JUnit), reading from external files(POI for excels) etc.

Ques.5. Which all browsers/drivers are supported by Selenium Webdriver?

Ans. Some commonly used browsers supported by selenium are-
Google Chrome - ChromeDriver
Firefox - FireFoxDriver
Internet Explorer - InternetExplorerDriver
Safari - SafariDriver
HtmlUnit (Headless browser) - HtmlUnitDriver

Android - Selendroid/Appium
IOS - ios-driver/Appium

Ques.6. Can we test APIs or web services using Selenium webdriver?

Ans. No selenium webdriver uses browser's native method to automate the web applications. Since web services are headless, so we cannot automate web services using selenium webdriver.

Ques.7. What are the testing type supported by Selenium WebDriver?

Ans. Selenium webdriver can be used for performing automated functional and regression testing.

Ques.8. What are various ways of locating an element in selenium?

Ans. The different locators in selenium are-
Id
XPath
cssSelector
className
tagName
name
linkText
partialLinkText

Ques.9. What is an XPath?

Ans. Xpath or XML path is a query language for selecting nodes from XML documents. XPath is one of the locators supported by selenium webdriver.

Ques.10. What is an absolute XPath?

Ans. An absolute XPath is a way of locating an element using an XML expression beginning from root node i.e. html node in case of web pages. The main disadvantage of absolute xpath is that even with slightest change in the UI or any element the whole absolute XPath fails.
Example - html/body/div/div[2]/div/div/div/div[1]/div/input

Ques.11. What is a relative XPath?

Ans. A relative XPath is a way of locating an element using an XML expression beginning from anywhere in the HTML document. There are different ways of creating relative XPaths which are used for creating robust XPaths (unaffected by changes in other UI elements).
Example - //input[@id='username']

Ques.12. What is the difference between single slash(/) and double slash(//) in XPath?

Ans. In XPath a single slash is used for creating XPaths with absolute paths beginning from root node.
Whereas double slash is used for creating relative XPaths.

Ques.13. Which XPath you will prefer to use? Why?

Normally we prefer to use Relative XPath.
Ralative Xpath can identify element evethou some UI changes happed, but can't identify by Absolute Xpath.

Ques.14. What is the difference between Absolute XPath and Relative XPath?

Absolute Xpath will traverse entire HTML from the root node /html.
Relative Xpath directly jump to node based on attribute specified.

Ques.15. How can we inspect the web element attributes in order to use them in different locators?

Ans. Using Chropath or developer tools we can inspect the specific web elements. Chropath is a plugin that provides xpaths and CSS Selectors. From automation perspective, "Right click on page inspect element" is used specifically for inspecting web-elements in order to use their attributes like id, class, name etc. in different locators.

Ques.16. How can we locate an element by only partially matching its attributes value in Xpath?

Ans. Using contains() method we can locate an element by partially matching its attribute's value. This is particularly helpful in the scenarios where the attributes have dynamic values with certain constant part.
xPath expression = //*[contains(@name,'user')]
The above statement will match the all the values of name attribute containing the word 'user' in them.

Ques.17. How can we locate elements using their text in XPath?

Ans. Using the text() method

Ques.18. How can we move to nth child element using XPath?

Ans. There are two ways of navigating to the nth element using XPath-

Using square brackets with index position
Example - div[2] will find the second div element.

Using position()
Example - div[position()=3] will find the third div element.

Ques.19. What is the syntax of finding elements by class using CSS Selector?

Ans. By .className we can select all the element belonging to a particluar class e.g. '.inputtext ' will select all elements having class ' inputtext '.

Ques.20. What is the syntax of finding elements by id using CSS Selector?

Ans. By #idValue we can select all the elements belonging to a particluar id e.g. '#u_0_n' will select the element having id - u_0_n.

Ques.21. How can we select elements by their attribute value using CSS Selector?

Ans. Using [attribute=value] we can select all the element belonging to a particluar attribute e.g. '[type=radio]' will select the element having attribute type of value ‘radio'.

Ques.22. What is fundamental difference between XPath and css selector?

Ans. The fundamental difference between XPath and css selector is using XPaths we can traverse up in the document i.e. we can move to parent elements. Whereas using CSS selector we can only move downwards in the document.

Ques.23. How can we launch different browsers in selenium webdriver?

Ans. By creating an instance of driver of a particular browser-

Ques.24. What is the use of driver.get("URL") and driver.navigate().to("URL") command? Is there any difference between the two?

Ans. Both driver.get("URL") and driver.navigate().to("URL") commands are used to navigate to a URL passed as parameter. There is no difference between the two commands.

Ques.25. How can we type text in a textbox element using selenium?

```
WebElement searchTextBox = driver.findElement(By.id("search"));
searchTextBox.sendKeys("searchTerm");
```

Ques.26. How can we clear a text written in a textbox?

Ans. Using clear() method we can delete the text written in a textbox.

```
driver.findElement(By.id("elementLocator")).clear();
```

Ques.27. How to check a checkBox in selenium?

Ans. The same click() method used for clicking buttons or radio buttons can be used for checking

checkbox as well.

Ques.28. How can we submit a form in selenium?

Ans. Using submit() method we can submit a form in selenium.

driver.findElement(By.id("form1")).submit();

Also, the click() method can be used for the same purpose.

Ques.29. Explain the difference between close and quit command.

Ans. driver.close() - Used to close the current browser having focusdriver. quit() - Used to close all the browser instances

Ques.30. How to switch between multiple windows in selenium?

Ans. Selenium
has driver.getWindowHandles() and driver.switchTo().window("{windowHandleName}") commands to work with multiple windows.
The getWindowHandles() command returns a list of ids corresponding to each window and on passing a particular window handle to driver.switchTo().window("{windowHandleName}") command we can switch control/focus to that particular window

for (String windowHandle : driver.getWindowHandles())
{
driver.switchTo().window(handle);

Ques.31. What is the difference between driver.getWindowHandle() and driver.getWindowHandles() in selenium?

Ans. driver.getWindowHandle() returns a handle of the current page (a unique identifier)Whereas driver.getWindowHandles() returns a set of handles of the all the pages available.

Ques.32. How can we move to a particular frame in selenium?

Ans. The driver.switchTo() commands can be used for switching to frames.

driver.switchTo().frame("{frameIndex/frameId/frameName}");

For locating a frame we can either use the index (starting from 0), its name or Id.

Ques.33. Can we move back and forward in browser using selenium?

Ans. Yes, using driver.navigate().back() and driver.navigate().forward() commands we can move backward and forward in a browser.

Ques.34. Is there a way to refresh browser using selenium?

Ans. There a multiple ways to refresh a page in selenium-
Using driver.navigate().refresh() command
Using sendKeys(Keys.F5) on any textbox on the webpage

Ques.35. How can we maximize browser window in selenium?

Ans. We can maximize browser window in selenium using following command-

```
driver.manage().window().maximize();
```

Ques.36. How can we fetch a text written over an element?

Ans. Using getText() method we can fetch the text over an element.

```
String text = driver.findElement("elementLocator").getText();
```

Ques.37. How can we find the value of different attributes like name, class, value of an element?

Ans. Using getAttribute("{attributeName}") method we can find the value of different attrbutes of an element e.g.-

```
String valueAttribute = driver.findElement(By.id("elementLocator")).getAttribute("value");
```

Ques.38. How to delete cookies in selenium?

Ans. Using deleteAllCookies() method-

```
driver.manage().deleteAllCookies();
```

Ques.39. What is an implicit wait in selenium?

Ans. An implicit wait is a type of wait which waits for a specified time while locating an element before throwing NoSuchElementException. By default selenium tries to find elements immediately when required without any wait. So, it is good to use implicit wait. This wait is applied to all the elements of the current driver instance.

```
driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
```

Ques.40. What is an explicit wait in selenium?

Ans. An explicit wait is a type of wait which is applied to a particular web element untill the expected condition specified is met.

```
WebDriverWait wait = new WebDriverWait(driver, 10);
WebElement element =
wait.until(ExpectedConditions.elementToBeClickable(By.id("elementId")));
```

Ques.41. What are some expected conditions that can be used in Explicit waits?

Ans. Some of the commonly used expected conditions of an element that can be used with expicit waits are-

elementToBeClickable(WebElement element or By locator)
visibilityOfElementLocated(By locator)
attributeContains(WebElement element, String attribute, String value)
alertIsPresent()
titleContains(String title)
titleIs(String title)
textToBePresentInElementLocated(By, String)

Ques.42. What is fluent wait in selenium?

Ans. A fluent wait is a type of wait in which we can also specify polling interval(intervals after which driver will try to find the element) along with the maximum timeout value.

```
Wait wait = new FluentWait(driver)
.withTimeout(20, SECONDS)
.pollingEvery(5, SECONDS)
.ignoring(NoSuchElementException.class);
WebElement textBox = wait.until(new Function()
{
public WebElement apply(WebDriver driver) {
return driver.findElement(By.id("textBoxId"));
} } );
```

Ques.43. What are the different keyboard operations that can be performed in selenium?

Ans. The different keyboard operations that can be performed in selenium are-
.sendKeys("sequence of characters") - Used for passing character sequence to an input or textbox element.
.pressKey("non-text keys") - Used for keys like control, function keys etc that are non-text.
.releaseKey("non-text keys") - Used in conjuntion with keypress event to simulate releasing a key from keyboard event.

Ques.44. What are the different mouse actions that can be performed?

Ans. The different mouse events supported in selenium are
click(WebElement element)
doubleClick(WebElement element)
contextClick(WebElement element)
moveToEelement(WebElement element)
dragAndDrop(source WebElement, target WebElement)

Ques.45. Write the code to double click an element in selenium?

```
Actions action = new Actions(driver);
WebElement element=driver.findElement(By.id("elementId"));
action.doubleClick(element).build().perform();
```

Ques.46. Write the code to right click an element in selenium?

```
Actions action = new Actions(driver);
WebElement element=driver.findElement(By.id("elementId")); action.contextClick(element).
build().perform();
```

Ques.47. How to mouse hover an element in selenium?

```
Actions action = new Actions(driver);
WebElement element=driver.findElement(By.id("elementId"));
action.moveToElement(element). build().perform();
```

Ques.48. How to fetch the current page URL in selenium?

Ans. Using getCurrentURL() command we can fetch the current page URL-

driver.getCurrentUrl();

Ques.49. How can we fetch title of the page in selenium?

Ans. Using driver.getTitle(); we can fetch the page title in selenium. This method returns a string containing the title of the webpage.

Ques.50. How can we fetch the page source in selenium?

Ans. Using driver.getPageSource(); we can fetch the page source in selenium. This method returns a string containing the page source

Ques.51. How to verify tooltip text using selenium?

Ans. Webelements have an attribute of type 'title'. By fetching the value of 'title' attribute we can verify the tooltip text in selenium.

String toolTipText = element.getAttribute("title");

Ques.52. How to locate a link using its text in selenium?

Ans. Using linkText() and partialLinkText() we can locate a link.
The difference between the two is linkText matches the complete string passed as parameter to the link texts. Whereas partialLinkText matches the string parameter partially with the link texts.

WebElement link1 = driver.findElement(By.linkText("pavantestingtools"));
WebElement link2 = driver.findElement(By.partialLinkText("testingtools"));

Ques.53. What are DesiredCapabilities in selenium webdriver?

Ans. Desired capabilities are a set of key-value pairs that are used for storing or configuring browser specific properties like its version, platform etc in the browser instances.

Ques.54. How can we find all the links on a web page?

Ans. All the links are of anchor tag 'a'. So by locating elements of tagName 'a' we can find all the links on a webpage.

List links = driver.findElements(By.tagName("a"));

Ques.55. What are some commonly encountered exceptions in selenium?

Ans. Some of the commonly seen exception in selenium are-
NoSuchElementException - When no element could be located from the locator provided.
ElementNotVisibleException - When element is present in the dom but is not visible.

NoAlertPresentException - When we try to switch to an alert but the targetted alert is not present.
NoSuchFrameException - When we try to switch to a frame but the targetted frame is not present.
NoSuchWindowException - When we try to switch to a window but the targetted window is not present.
TimeoutException - When a command execution gets timeout.
InvalidElementStateException - When the state of an element is not appropriate for the desired action.
NoSuchAttributeException - When we are trying to fetch an attribute's value but the attribute is not correct
WebDriverException - When there is some issue with driver instance preventing it from getting launched.

Ques.56. How can we capture screenshots in selenium?

Ans. Using getScreenshotAs method of TakesScreenshot interface we can take the screenshots in selenium.

```
File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(scrFile, new File("D:\\testScreenShot.jpg"));
```

Ques.57. How to handle dropdowns in selenium?

Ans. Using Select class-

```
Select countriesDropDown = new Select(driver.findElement(By.id("countries")));
dropdown.selectByVisibleText("India"); //or using index of the option starting from 0
dropdown.selectByIndex(1); //or using its value attribute
dropdown.selectByValue("Ind");
```

Ques.58. How to check which option in the dropdown is selected?

Ans. Using isSelected() method we can check the state of a dropdown's option.

```
Select countriesDropDown = new Select(driver.findElement(By.id("countries")));
dropdown.selectByVisibleText("India"); //returns true or false value
System.out.println(driver.findElement(By.id("India")).isSelected());
```

Ques.59. How can we check if an element is getting displayed on a web page?

Ans. Using isDisplayed method we can check if an element is getting displayed on a web page.

```
driver.findElement(By locator).isDisplayed();
```

Ques.60. How can we check if an element is enabled for interaction on a web page?

Ans. Using isEnabled method we can check if an element is enabled or not.

driver.findElement(By locator).isEnabled();

Ques.61. What is the difference between driver.findElement() and driver.findElements() commands?

Ans. findElement() returns a single WebElement (found first) based on the locator passed as parameter. Whereas findElements() returns a list of WebElements, all satisfying the locator value passed.
Syntax of findElement():WebElement textbox = driver.findElement(By.id("textBoxLocator"));
Syntax of findElements():List elements = element.findElements(By.id("value"));
Another difference between the two is- if no element is found then findElement() throws NoSuchElementException whereas findElements() returns a list of 0 elements.

Ques.62. Explain the difference between implicit wait and explicit wait.?

Ans. An implicit wait, while finding an element waits for a specified time before throwing NoSuchElementException in case element is not found. The timeout value remains valid throughout the webDriver's instance and for all the elements.

driver.manage().timeouts().implicitlyWait(180, TimeUnit.SECONDS);

Whereas, Explicit wait is applied to a specified element only-

WebDriverWait wait = new WebDriverWait(driver, 5);
wait.until(ExpectedConditions.presenceOfElementLocated(ElementLocator));

Ques.63. How can we handle window UI elements and window POP ups using selenium?

Ans. Selenium is used for automating Web based application only(or browsers only). For handling window GUI elements we can use AutoIT or Sikuli.

Ques.64. What is Robot API?

Ans. Robot API is used for handling Keyboard or mouse events. It is generally used to upload files to the server in selenium automation

```
Robot robot = new Robot(); //Simulate enter key action
robot.keyPress(KeyEvent.VK_ENTER);
```

Ques.65. How to do file upload in selenium?

Ans. File upload action can be performed in multiple ways-
Using element.sendKeys("path of file") on the webElement of input tag and type file i.e. the elements should be like…

Using Robot API.
Using AutoIT.
Using Sikuli

Ques.66. How to handle HTTPS website in selenium? or How to accept the SSL untrusted connection?

Ans. Using profiles in firefox we can handle accept the SSL untrusted connection certificate. Profiles are basically set of user preferences stored in a file.

Firefox

```
FirefoxProfile profile = new FirefoxProfile();
profile.setAcceptUntrustedCertificates(true);
profile.setAssumeUntrustedCertificateIssuer(false);
WebDriver driver = new FirefoxDriver(profile);
```

IE

```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
System.setProperty("webdriver.ie.driver","IEDriverServer.exe");
WebDriver driver = new InternetExplorerDriver(capabilities);
```

Chrome

```
DesiredCapabilities handlSSLErr = DesiredCapabilities.chrome ()
handlSSLErr.setCapability (CapabilityType.ACCEPT_SSL_CERTS, true)
WebDriver driver = new ChromeDriver (handlSSLErr);
```

Ques.67 How to do drag and drop in selenium?

Using Action class, drag and drop can be performed in selenium. Sample code-

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

act.clickAndHold(source Element).moveToElement(target Element).release().build().perform();

OR

act.dragAndDrop(source Element, target Element).build().perform();
```

Ques.68. How to execute javascript in selenium?

Ans. JavaScript can be executed in selenium using JavaScriptExecuter. Sample code for javascript execution-

```
 JavascriptExecutor js = ((JavascriptExecutor) driver);
js.executeScript("{Java script code }");
```

Ques.69. How to handle alerts in selenium?

Ans. In order to accept or dismiss an alert box the alert class is used. This requires first switching to the alert box and than using accept() or dismiss() command as the case may be.

```
Alert alert = driver.switchTo().alert(); //To accept the alert
alert.accept();

Alert alert = driver.switchTo().alert(); //To cancel the alert box
alert.dismiss();
```

Ques.70. What is HtmlUnitDriver?

Ans. HtmlUnitDriver is the fastest WebDriver. Unlike other drivers (FireFoxDriver, ChromeDriver etc), the HtmlUnitDriver is non-GUI, while running no browser gets launched.

Ques.71. How to handle hidden elements in Selenium webDriver?

Ans. Using javaScript executor we can handle hidden elements-

```
(JavascriptExecutor(driver))
.executeScript("document.getElementsByClassName(ElementLocator).click();");
```

Ques.72. What is Page Object Model or POM?

Ans. Page Object Model(POM) is a design pattern in selenium. POM helps to create a framework for maintaining selenium scripts.
In POM for each page of the application a class is created having the web elements belonging to the page and methods handling the events in that page.

The test scripts are maintained in separate files and the methods of the page object files are called from the test scripts file.

Ques.73. What are the advantages of POM?

Ans. The advantages are POM are-
Using POM we can create an Object Repository, a set of web elements in separate files along with their associated functions. Thereby keeping code clean.
For any change in UI(or web elements) only page object files are required to be updated leaving test files unchanged.
It makes code reusable and maintainable.

Ques.74. What is Page Factory?

Ans. Page factory is an implementation of Page Object Model in selenium. It provides @FindBy annotation to find web elements and PageFactory.initElements() method to initialize all web elements defined with @FindBy annotation.

```
public class SamplePage
{
WebDriver driver;
@FindBy(id="search")
WebElement searchTextBox;
@FindBy(name="searchBtn")
 WebElement searchButton;

//Constructor public samplePage(WebDriver driver)
{
this.driver = driver; //initElements method to initialize all elements
PageFactory.initElements(driver, this);
}
 //Sample method
public void search(String searchTerm)
{
 searchTextBox.sendKeys(searchTerm); searchButton.click();
```

Ques.75. What is an Object repository?

Ans. An object repository is centralized location of all the objects or WebElements of the test scripts.
In selenium we can create object repository using Page Object Model and Page Factory design patterns

Ques.76. What is a data driven framework?

Ans. A data driven framework is one in which the test data is put in external files like csv, excel etc separated from test logic written in test script files. The test data drives the test cases, i.e. the test methods run for each set of test data values.

Ques.77. What is a keyword driven framework?

Ans. A keyword driven framework is one in which the actions are associated with keywords and kept in external files e.g. an action of launching a browser will be associated with keyword - launchBrowser(), action to write in a textbox with keyword - writeInTextBox(webElement, textToWrite) etc.
The code to perform the action based on a keyword specified in external file is implemented in the framework itself.

Ques.78. What is a hybrid framework?

Ans. A hybrid framework is a combination of one or more frameworks. Normally it is associated with combination of data driven and keyword driven frameworks where both the test data and test actions are kept in external files(in the form of table).

Ques.79. What is selenium Grid?

Ans. Selenium grid is a tool that helps in distributed running of test scripts across different machines having different browsers, browser version, platforms etc in parallel. In selenium grid there is hub that is a central server managing all the distributed machines known as nodes.

Ques.80. What are some advantages of selenium grid?

Ans. The advantages of selenium grid are-
It allows running test cases in parallel thereby saving test execution time.
Multi browser testing is possible using selenium grid by running the test on machines having different browsers.
It is allows multi-platform testing by configuring nodes having different operating systems.

Ques.81. What is a hub in selenium grid?

Ans. A hub is server or a central point in selenium grid that controls the test executions on the different machines.

Ques.82. What is a node in selenium grid?

Ans. Nodes are the machines which are attached to the selenium grid hub and have selenium instances running the test scripts. Unlike hub there can be multiple nodes in selenium grid.

Ques.83. Explain the line of code Webdriver driver = new FirefoxDriver();

Ans. In the line of code Webdriver driver = new FirefoxDriver();
'WebDriver' is an interface and we are creating an object of type WebDriver instantiating

an object of FirefoxDriver class.

Ques.84 What is the purpose of creating a reference variable- 'driver' of type WebDriver instead of directly creating a FireFoxDriver object or any other driver's reference in the statement Webdriver driver = new FirefoxDriver();?

Ans. By creating a reference variable of type WebDriver we can use the same variable to work with multiple browsers like ChromeDriver, IEDriver etc.

Ques.85. What is testNG?

Ans. TestNG(NG for Next Generation) is a testing framework that can be integrated with selenium or any other automation tool to provide multiple capabilities like assertions, reporting, parallel test execution etc.

Ques.86. What are some advantages of testNG?

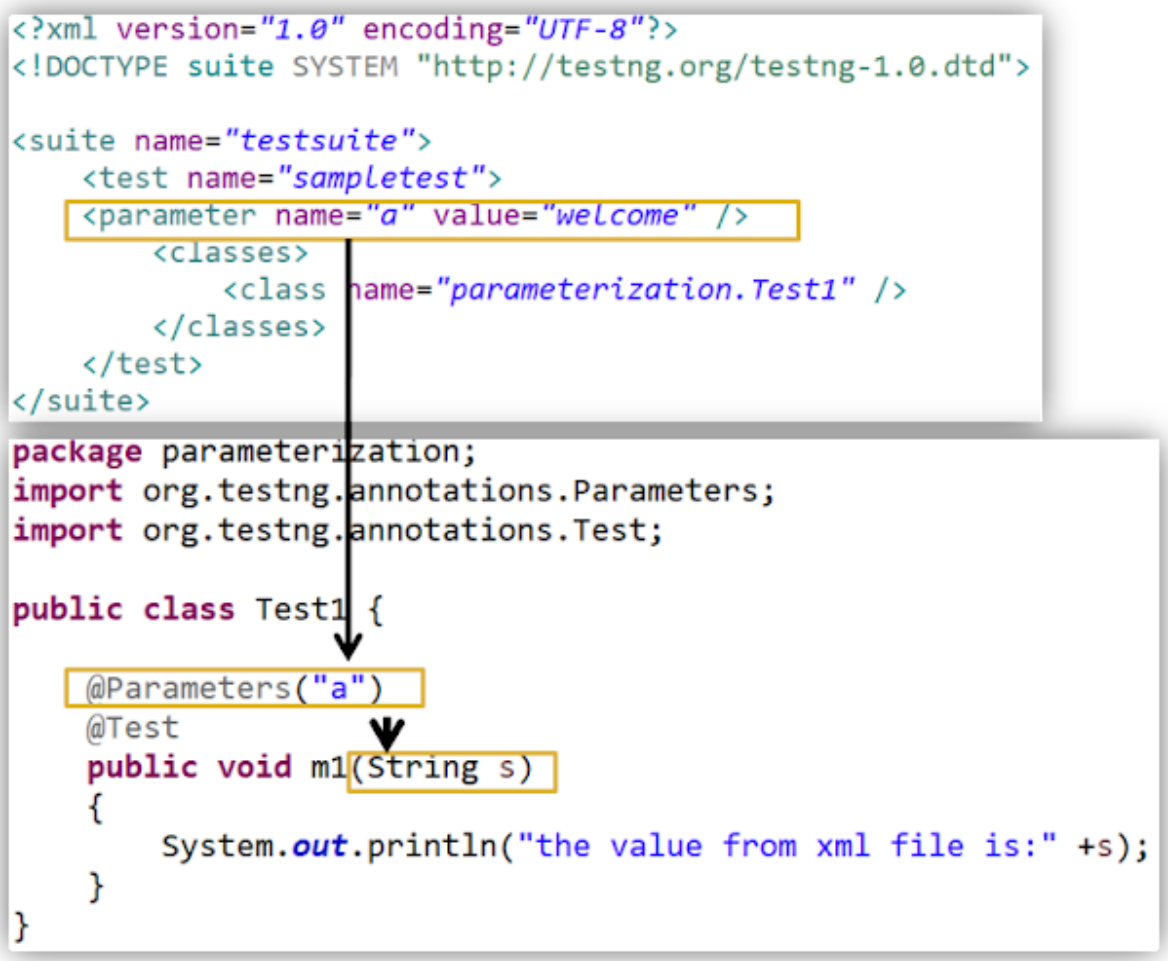
Ans. Following are the advantages of testNG:
TestNG provides different assertions that helps in checking the expected and actual results.
It provides parallel execution of test methods.
We can define dependency of one test method over other in TestNG.
We can assign priority to test methods in selenium.
It allows grouping of test methods into test groups.
It allows data driven testing using @DataProvider annotation.
It has inherent support for reporting.
It has support for parameterizing test cases using @Parameters annotation.

Ques.87. What is the use of testng.xml file?

Ans. testng.xml file is used for configuring the whole test suite.
In testng.xml file we can create test suite, create test groups, mark tests for parallel execution, add listeners and pass parameters to test scripts.
Later this testng.xml file can be used for triggering the test suite.
Ques.88. How can we pass parameter to test script using testNG?
Ans. Using @Parameter annotation and 'parameter' tag in testng.xml we can pass parameters to the test script.

Ques.88. How can we pass parameter to test script using testNG?

Ans. Using @Parameter annotation and 'parameter' tag in testng.xml we can pass parameters to the test script.



Ques.89. How can we create data driven framework using testNG?

Ans. Using @DataProvider we can create a data driven framework in which data is passed to the associated test method and multiple iteration of the test runs for the different test data values passed from the @DataProvider method.
The method annotated with @DataProvider annotation return a 2D array of object.


```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class DataProviderExample {

    WebDriver driver;

    @BeforeClass
    void setup()
    {
        System.setProperty("webdriver.chrome.driver","C://Drivers/chromedriver_win32/chromedriver.exe");
        driver=new ChromeDriver();
    }

    @Test(dataProvider="users")
    void loginTest(String uname,String pwd)
    {
        driver.get("http://newtours.demoaut.com/");
        driver.findElement(By.name("userName")).sendKeys(uname);
        driver.findElement(By.name("password")).sendKeys(pwd);
        driver.findElement(By.name("login")).click();
        Assert.assertEquals(driver.getTitle(), "Find a Flight: Mercury Tours:");
    }

    @DataProvider(name="users")
    String [][] logindata()
    {
        String data[][]={{"mercury","mercury"},{"asasa","mercury1"},{"mercury2","mercury2"} };
        return data;
    }

    @AfterClass
    void closebrowser()
    {
    driver.quit();
    }

}
```

Ques.90. What is the use of TestNG Listeners?

Ans. TestNG provides us different kind of listeners using which we can perform some action in case an event has triggered.
Usually testNG listeners are used for configuring reports and logging.
One of the most widely used lisetner in testNG is ITestListener interface and TestListenerAdapter Class.
It has methods like onTestSuccess, onTestFailure, onTestSkipped etc.
We need to implement this interface creating a listener class of our own.
Ques.91. What is the use of @Listener annotation in TestNG?
We need to implement ITestListener interface by creating a listener class of our own. After that using the @Listener annotation, we can use specify that for a particular test class, our customized listener class should be used.

Ques.91. What is the use of @Listener annotation in TestNG?

Ans. We need to implement ITestListener interface by creating a listener class of our own. After that using the @Listener annotation, we can use specify that for a particular test class,

our customized listener class should be used.

```
package testnglisteners;

import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;

public class Mylisteners implements ITestListener {
    @Override
    public void onTestStart(ITestResult result) {
        System.out.println(" test is started");
    }

    @Override
    public void onTestSuccess(ITestResult result) {
        System.out.println(" test is passed");
    }

    @Override
    public void onTestFailure(ITestResult result) {
        System.out.println("test is failed");
    }

    @Override
    public void onTestSkipped(ITestResult result) {
        System.out.println("test is skipped");
    }

    @Override
    public void onTestFailedButWithinSuccessPercentage(ITestResult result) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onStart(ITestContext context) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onFinish(ITestContext context) {
        // TODO Auto-generated method stub

    }
}
```

```java
package testnglisteners;

import org.testng.Assert;
import org.testng.annotations.Listeners;
import org.testng.annotations.Test;

@Listeners(testnglisteners.Mylisteners.class)
public class LoginTest {

    @Test
    void setup()
    {
        Assert.fail();
    }

    @Test
    void loginByEmail()
    {
        Assert.assertTrue(true);
    }

    @Test(dependsOnMethods={"setup"})
    void loginByFacebook()
    {
        Assert.assertTrue(true);
    }
}
```

Ques.92. How can we make one test method dependent on other using TestNG?

Ans. Using dependsOnMethods parameter inside @Test annotation in testNG we can make one test method run only after successful execution of dependent test method.

@Test(dependsOnMethods = { "preTests" })

Ques.93. How can we set priority of test cases in TestNG?

Ans. Using priority parameter in @Test annotation in TestNG we can define priority of test cases. The default priority of test when not specified is integer value 0. Example:

@Test(priority=1)

Ques.94. What are commonly used TestNG annotations?

Ans. The commonly used TestNG annotations are-
@Test

@BeforeMethod
@AfterMethod
@BefoerClass
@AfterClass
@BeforeTest
@AfterTest
@BeforeSuite
@AfterSuite

Ques.95. What are some common assertions provided by testNG?

Ans. Some of the common assertions provided by testNG are-
assertEquals(String actual, String expected, String message) - (and other overloaded data type in parameters)
assertNotEquals(double data1, double data2, String message) - (and other overloaded data type in parameters)
assertFalse(boolean condition, String message)
assertTrue(boolean condition, String message)
assertNotNull(Object object)
fail(boolean condition, String message)
true(String message)

Ques.96. How can we run test cases in parallel using TestNG?

Ans. In order to run the tests in parallel just add these two key value pairs in suite-
parallel="{methods/tests/classes}"
thread-count="{number of thread you want to run simultaneously}".

```java
import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class CrossBrowserTest {

    WebDriver driver = null;

    @Parameters("browser")
    @Test
    public void launchBrowser(String br) {
        if (br.equals("FF")) {
            // Firefox Browser
            System.setProperty("webdriver.gecko.driver", "C:///Drivers/geckodriver-v0.19.1-win64/geckodriver.exe");
            driver = new FirefoxDriver();
        }

        else if (br.equals("IE")) {
            System.setProperty("webdriver.ie.driver", "C:///Drivers/IEDriverServer_Win32_3.7.0/IEDriverServer.exe");
            driver = new InternetExplorerDriver();
        }

        else if (br.equals("CH")) {
            System.setProperty("webdriver.chrome.driver", "C:///Drivers/chromedriver_win32/chromedriver.exe");
            driver = new ChromeDriver(); // opens the browser
        }

        driver.get("http://newtours.demoaut.com/");

    }

    @Test
    public void verifyTitle() {
        Assert.assertEquals(driver.getTitle(), "Welcome: Mercury Tours");
    }

    /*
     * @Test public void registration() { //selenium code for registration }
     */

    @AfterClass
    public void closeBrowser() {
        driver.quit();
    }

}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name="paralleltesting" parallel="tests" thread-count="5">

    <test name="FirefoxTest">
    <parameter name="browser" value="FF" />
        <classes>
            <class name="parameterization.CrossBrowserTest" />
        </classes>
    </test>


    <test name="ChromeTest">
    <parameter name="browser" value="CH" />
        <classes>
            <class name="parameterization.CrossBrowserTest" />
        </classes>
    </test>


    <test name="IETest">
    <parameter name="browser" value="IE" />
        <classes>
            <class name="parameterization.CrossBrowserTest" />
        </classes>
    </test>

</suite>
```

Ques.97. Name an API used for reading and writing data to excel files.

Ans. Apache POI API and JXL(Java Excel API) can be used for reading, writing and updating excel files.

Ques.98. Name an API used for logging in Java.

Ans. Log4j is an open source API widely used for logging in Java.
It supports multiple levels of logging like - ALL, DEBUG, INFO, WARN, ERROR, TRACE and FATAL.

Ques.99. What is the use of logging in automation?

Ans. Logging helps in debugging the tests when required and also provides a storage of test's runtime behaviour.

Ques.100. What is InvocationCount in TestNG?

This is a TestNG attribute that defines number of times a test method should be invoked or executed before executing any other test method.

```java
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;

public class invocationCount {

    @Test(invocationCount = 5)
    public void getTitle() {
        System.setProperty("webdriver.chrome.driver", "C:/Drivers/chromedriver_win32/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.get("http://www.pavantestingtools.com/");
        driver.manage().window().maximize();
        System.out.println("Website Title: "+driver.getTitle());
        driver.quit();
    }

    @Test
    public void secondTest() {
        System.out.println("This will be executed at the end");
    }
}
```