***UFT Assignment Questions***

### **1. Introduction to UFT:**

**Q1:** What is UFT (Unified Functional Testing)? How is it different from other test automation tools like Selenium or QTP?

UFT, or Unified Functional Testing, is an automation tool by Micro Focus used for testing software applications. It’s mainly used to automate functional, regression, and GUI testing. UFT supports both desktop and web applications and can work with different technologies like Java, .NET, SAP, and Oracle.

The difference between UFT and other tools is that UFT uses a Visual Basic scripting language for automation, while Selenium, for example, supports multiple languages like Java, Python, and C#. Also, UFT provides a built-in IDE, recording, and debugging features which makes it beginner-friendly. On the other hand, Selenium is open-source and mainly used for web applications, while QTP was UFT’s older version. UFT is more advanced than QTP as it now supports both API and GUI testing together.

**Q2:** List the key features of UFT. Explain how it supports functional, regression, and GUI testing.

Some of the key features of UFT are:

* Keyword and Script-based testing
* Record and Playback functionality
* Integration with ALM (Application Lifecycle Management)
* Support for multiple technologies like Java, .NET, SAP, Oracle
* Built-in debugging and reporting tools

UFT supports functional testing by checking whether each feature of an application works according to requirements. It helps in regression testing by rerunning tests on an updated application to ensure new changes haven’t broken existing features. For GUI testing, UFT can interact with buttons, text boxes, and menus on the screen by recognizing them as objects. This makes it easy to automate tasks and verify the look and feel of the application’s interface.

**Q3:** What are the different types of objects that UFT can recognize? Give examples of each type.

In UFT, objects are anything you interact with on a software application like buttons, text fields, or menus. There are mainly three types of objects:

1. **Standard Objects** — These are the usual objects like buttons, text boxes, combo boxes, and links.  
   *Example:* A login button on a website.
2. **Custom Objects** — These are objects specially created by developers and may not behave like standard objects.  
   *Example:* A graphical dial in a banking application.
3. **Runtime Objects** — These are objects created while the application is running.  
   *Example:* A message box that appears only after clicking a certain button.

UFT identifies these objects during test creation and adds them to the object repository for use during test execution.

### **2. Creating and Running a Basic Test in UFT:**

**Q4:** Create a simple test in UFT to open the Notepad application, type a text message, and save the file. Include the steps to record and run the test.

**Q5:** Write a simple UFT script to open a web browser, navigate to a website (e.g., www.google.com), and perform a Google search.

SystemUtil.Run "iexplore.exe", "http://www.google.com"

Browser("Google").Page("Google").WebEdit("q").Set "UFT tutorial"

Browser("Google").Page("Google").WebButton("Google Search").Click

Explanation:

SystemUtil.Run opens Internet Explorer and navigates to Google.

WebEdit("q") identifies the search box and inputs text.

WebButton("Google Search") clicks the search button.

### **3. Object Repository and Object Identification:**

**Q6:** What is an object repository in UFT? Explain the difference between "Local Object Repository" and "Shared Object Repository."

An Object Repository (OR) in UFT is a storage location where information about objects used in your tests is saved. When you record or manually create a test, UFT identifies objects from the application and stores their properties in the OR so it can interact with them during test execution.

**Types:**

* **Local Object Repository (LOR):**  
  This repository is specific to a particular test. It’s automatically created when you record a test and the objects stay within that test.
* **Shared Object Repository (SOR):**  
  This is a central repository that can be linked to multiple tests. It helps in reusing objects across different tests, making maintenance easier.

**Q7:** Explain the concept of "Object Identification" in UFT. How does UFT recognize objects on the application being tested?

Object Identification in UFT is the process of recognizing and storing the properties of objects on a software application so UFT can interact with them during automation. Every object on a screen, like a button or text field, has unique properties such as its name, type, or position.

When you record a test or manually add an object, UFT captures these properties and stores them in the Object Repository. During test execution, UFT uses these stored properties to locate and interact with the correct object.

If multiple objects have similar properties, UFT uses a method called Smart Identification to find the right one based on additional properties. This makes test scripts reliable even when there are small changes in the application’s UI.

### **4. Checkpoints and Verification:**

* **Q10:** What are checkpoints in UFT? Write a script to add a "Text Checkpoint" to verify that a specific text appears on a web page.

**Q11:** Explain the difference between "Standard Checkpoints" and "Database Checkpoints" in UFT. Give an example of when you would use each.

A Standard Checkpoint is used to verify the properties of objects in your application, like checking if a button is enabled, a text label is correct, or an image exists.  
Example: Checking if a “Login Successful” message appears after login.

A Database Checkpoint, on the other hand, is used to verify the data stored in a database. You can check whether values in a database table match the expected values.

Example: After submitting a form, you can use a database checkpoint to confirm if the entered data was successfully saved in the database table.

So, Standard Checkpoints are for screen elements and Database Checkpoints are for backend data verification.

**Q12:** How can you handle dynamic objects using UFT? Explain with an example of handling dynamic buttons that change text based on user interactions.

In UFT, dynamic objects are those whose properties like name, ID, or text keep changing every time the application runs. To handle them, we can use regular expressions in the Object Repository or create Descriptive Programming scripts.

Example:  
If a button text changes from “Submit” to “Submit Now” dynamically:

Set btn = Browser("DemoApp").Page("DemoApp").WebButton("text:=Submit.\*")

btn.Click

Explanation:  
Here, text:=Submit.\* uses a regular expression. The .\* means “anything after Submit”, so it can handle “Submit”, “Submit Now”, “Submit Form” etc. This way, UFT can still find and click the button even if its text changes.

### **5. Parameterization:**

**Q13:** What is parameterization in UFT? Why is it important for automating tests? Demonstrate how to parameterize a test using input data (e.g., user credentials for a login page).

**Parameterization** in UFT means replacing hardcoded values in a test with variables (parameters) so the same test can run multiple times with different data. It’s important because it helps in data-driven testing, where you can test the same scenario with different sets of inputs, saving time and improving test coverage.

**Example:**  
For a login test, instead of typing the username and password in the script, you can link them to a data table:

Browser("DemoApp").Page("DemoApp").WebEdit("Username").Set DataTable("UserName", dtGlobalSheet)

Browser("DemoApp").Page("DemoApp").WebEdit("Password").Set DataTable("Password", dtGlobalSheet)

Now, you can store multiple usernames and passwords in the data table and UFT will use them in different test iterations.

**Q14:** Create a test that accepts input parameters (e.g., username and password) from an Excel file and performs a login using that data.

RowCount = DataTable.GetSheet("Global").GetRowCount

For i = 1 To RowCount

DataTable.SetCurrentRow i

Browser("DemoApp").Page("DemoApp").WebEdit("Username").Set DataTable("UserName", dtGlobalSheet)

Browser("DemoApp").Page("DemoApp").WebEdit("Password").Set DataTable("Password", dtGlobalSheet)

Browser("DemoApp").Page("DemoApp").WebButton("Login").Click

Next

**Q15:** What are the different types of parameters available in UFT (e.g., test, action, and data table parameters)? Explain their use with examples.

In UFT, there are mainly three types of parameters:

1. Test Parameters:  
   These are global parameters that can be used throughout the entire test.  
   *Example:* A parameter for environment URL.
2. Action Parameters:  
   These are local to specific actions within a test. You can pass data into and out of actions.  
   *Example:* Passing a customer name from one action to another.
3. Data Table Parameters:  
   These are values taken from the UFT data table (like an Excel sheet) during test execution. It allows running the same test with multiple sets of data.  
   *Example:* A list of usernames and passwords.

Use Case:  
If you have a login test and want to run it with 5 different users:

* Use a Data Table Parameter for usernames and passwords.
* Use a Test Parameter to store the URL.
* Use Action Parameters if you want to pass a specific message between actions.

### **6. Actions and Function Libraries:**

**Q16:** What is an action in UFT? How does it help in organizing your test scripts? Create an example of a reusable action for logging into a web application.

In UFT, an Action is a part of a test that contains a set of steps to perform a specific task. Actions help break down large tests into smaller, manageable pieces. This makes the test easier to read, debug, and reuse. UFT provides Reusable Actions, which can be called from multiple tests or actions, and Non-Reusable Actions, which are used only within the same test.

**Q17:** Explain the concept of "Function Libraries" in UFT. How do you create and associate a function library with your test?

A Function Library in UFT is a VBScript file (.vbs) that stores reusable functions used in multiple tests. It helps keep your scripts clean by moving repetitive code like login or calculations into a separate library.

To create a Function Library:

* Go to File > New > Function Library.
* Write your functions and save the file with a .vbs extension.
* To link it with your test, go to Resources > Associate Function Library, and select the library file.

This makes your tests modular and easy to maintain. If you need to update a function, you only do it in one place instead of editing every test script.

**Q18:** Write a simple function in a UFT function library that accepts two numbers as inputs and returns their sum. Call this function from your test script.

### **7. Descriptive Programming:**

**Q19:** What is Descriptive Programming in UFT, and when would you use it? Write a UFT script using descriptive programming to click a button on a webpage (e.g., a "Submit" button).

Descriptive Programming (DP) is a technique in UFT where you directly describe an object’s properties in the script instead of using the Object Repository. You use it when:

* Objects are dynamic.
* Objects are not stored in the repository.
* Quick one-time test creation.

Browser("title:=DemoApp").Page("title:=DemoApp").WebButton("name:=Submit").Click

**Q20:** Explain the syntax for Descriptive Programming in UFT. Write a script that uses descriptive programming to interact with a web element based on its properties (e.g., link text, tagname, etc.).

ObjectType("property:=value").Method

Browser("title:=DemoApp").Page("title:=DemoApp").Link("text:=Click Here", "html tag:=A").Click

This identifies a link using its text and HTML tag and performs a click without referring to the Object Repository.

**Q21:** How does UFT handle dynamic objects with Descriptive Programming? Provide an example using a dynamic link or button.

When dealing with dynamic objects whose properties change often (like IDs or names), UFT can use regular expressions in Descriptive Programming.

Example:

Browser("title:=DemoApp").Page("title:=DemoApp").Link("text:=Order.\*").Click

This clicks on any link starting with "Order" like "Order Now" or "Order 123". The .\* means "anything after Order", helping UFT handle dynamic elements flexibly.

**Q22: Why is synchronization important in UFT? What are the different synchronization techniques you can use to make sure your script waits for an element to be available?**

**Synchronization** ensures that UFT waits for the application to be ready before performing an action. Without it, scripts may fail if they try to interact with elements before they load.

**Synchronization Techniques:**

1. **Wait Property:** Waits until an object’s property meets a condition.
2. **Sync Method:** Waits for the browser to finish loading.
3. **Wait Method:** Pauses the test for a specified time.
4. **Exist Method:** Waits until an object exists before proceeding.
5. **Global Synchronization Timeout:** Set in **Test Settings** as the maximum time UFT waits for an object.

Good synchronization makes your tests stable, especially for slow applications.

**Q23: Write a script that uses the Sync method and Wait method to ensure UFT waits for a page to load before performing actions like clicking a button.**

**Simple Script:**

Browser("title:=DemoApp").Page("title:=DemoApp").Sync

Wait(3)

Browser("title:=DemoApp").Page("title:=DemoApp").WebButton("name:=Submit").Click

The Sync method waits for the browser to load. Wait(3) pauses the test for 3 seconds before clicking the "Submit" button.

**Q24: How would you handle synchronization issues when testing a slow application or a page with dynamic content?**

When an application is slow or has dynamic content, you can handle synchronization issues by:

* Using Sync for web pages.
* Using WaitProperty to wait until an object is enabled or visible.
* Using Exist method with a timeout to wait for an object’s appearance.
* Increasing the Global Synchronization Timeout from Test Settings.
* Using Wait method as a last resort for a fixed delay.

A balanced use of these ensures that your test remains stable and efficient without adding unnecessary waits.

**Q25: How can you add exception handling in UFT to handle pop-ups or alerts that appear unexpectedly during the test execution?**

In UFT, you can handle unexpected pop-ups using:

* **Recovery Scenarios:** Automatically handles exceptions like pop-ups.
* **On Error Resume Next:** Skips the error and continues test execution.
* **Regular Checks using Exist:** Check if a pop-up exists before interacting.

If Dialog("text:=Warning").Exist(2) Then

Dialog("text:=Warning").WinButton("text:=OK").Click

End If

This checks if a dialog box appears within 2 seconds and clicks OK if it does.

**Q26: Explain how UFT generates test results. How do you view and analyze the test results after running a test in UFT?**

After running a test, UFT automatically generates a Test Results Report. This report shows:

* The test name, time, and status (pass/fail)
* Details of each step’s execution
* Any errors or warnings encountered
* Screenshots (if enabled) for failed steps

You can view it by going to **File > Test Results Viewer** or clicking the Test Results icon.

The report helps analyze what went wrong, which step failed, and the reason, making it easy to debug and fix issues.

**Q27: What is the difference between the "Test Results" tab and the "Run-Time Data Table" in UFT? How would you use them to debug a failing test?**

**Test Results Tab:**

Shows a summary and detailed step-by-step result of a test run — which steps passed, failed, or had warnings.

Run-Time Data Table:  
Captures and displays the data used in each test iteration during execution. It reflects actual data values as they were at runtime.

Usage:  
If a test fails, first check the Test Results tab to see where it failed. Then, check the Run-Time Data Table to verify if the correct data was used in that iteration. Together, they help in understanding and fixing errors effectively.