UFT Assignment

Ans1:
UFT (Unified Functional Testing), previously known as QTP (QuickTest Professional), is an automated functional testing tool by Micro Focus. It allows testers to create automated test scripts for desktop, web, and mobile applications. UFT uses VBScript as its scripting language and supports both GUI and API testing. Compared to Selenium, which is open-source and supports only web apps, UFT is a commercial tool with a built-in IDE and strong integration for both front-end and back-end testing. Unlike QTP, which focused mainly on GUI testing, UFT provides more advanced features and supports modern technologies.

Ans2:
UFT offers features like keyword and scripting views, record and playback, object repositories, checkpoints, integration with ALM, data-driven testing, and cross-platform testing. It supports functional testing by validating features against requirements. For regression testing, it can rerun automated tests to check if changes broke existing functionality. GUI testing is handled through object recognition, allowing interaction with buttons, input fields, and other UI elements to validate the user interface's behavior.

Ans3:
UFT recognizes different object types like Standard Objects (e.g., buttons, checkboxes), Web Objects (e.g., web edit, web link), and Custom Objects defined by developers. For example, a login button is a WebButton, a textbox is a WebEdit, and a dropdown is a WebList. UFT can also interact with ActiveX controls, Java objects, and even SAP or Siebel application elements if proper add-ins are installed.

Ans4:
To create a test in UFT that opens Notepad, types a message, and saves the file, first launch UFT, start a new test, and click “Record.” Open Notepad manually and type your message. Use File > Save to store the text. Stop the recording. The generated script will include all actions (e.g., launching Notepad, typing, clicking menu). You can replay it by clicking “Run.” UFT captures actions based on Windows objects if the proper add-in is used.

Ans5:
To write a simple UFT script that opens Google and performs a search:

```
SystemUtil.Run "iexplore.exe", "http://www.google.com"
Browser("Google").Page("Google").WebEdit("q").Set "UFT Testing"
Browser("Google").Page("Google").WebButton("Google Search").Click
```

This script launches Internet Explorer, enters text into the search bar, and clicks the search button. You may need to use Object Repository or Descriptive Programming based on your setup.

Ans6:
An Object Repository in UFT is a storage location for all objects that UFT interacts with in a test. The Local Object Repository stores objects for a specific action, while the Shared Object Repository (also called Global Repository) is reusable across multiple tests or actions. Shared repositories help maintain consistency and reduce duplication when testing large applications with many common objects.

Ans7:
Object Identification in UFT is the process through which UFT recognizes elements in the application using their properties. UFT uses a combination of mandatory, assistive, and ordinal properties to uniquely identify each object. If multiple objects have the same properties, it uses an ordinal identifier like "index" to distinguish between them. This is key to making sure UFT interacts with the correct object during test execution.

Ans8:
Checkpoints in UFT are used to compare the current value of an application object with the expected value. A "Text Checkpoint" verifies if specific text appears on a page. Example script:

```
Browser("Google").Page("Google").Check CheckPoint("SearchText")
```

This assumes you’ve inserted a text checkpoint on the page. During playback, UFT will compare the expected text with what’s on the page and log a pass/fail result.

Ans9:
Standard Checkpoints check properties like text, enabled status, or existence of UI elements (e.g., "Is this button visible?"). Database Checkpoints verify data stored in databases by connecting to the DB and comparing expected vs actual values. For example, use a Standard Checkpoint to check if a label is correct and a Database Checkpoint to verify if user login data is stored correctly in the database table.

Ans10:
To handle dynamic objects in UFT, you can use regular expressions in object properties or Descriptive Programming. For example, if a button changes its label to “Submit Order” or “Submit Request,” you can define the property using a pattern:

```
Browser("app").Page("app").WebButton("text:=Submit.*").Click
```

This helps UFT locate the button regardless of the exact label, as long as it starts with “Submit.”

Ans11:
Standard Checkpoints are used to verify properties of UI objects like buttons, text boxes, and images. They check things like value, visibility, or tooltip text. Database Checkpoints are used to verify data in a database by connecting to it and running SQL queries. For example, you’d use a Standard Checkpoint to confirm a button's label and a Database Checkpoint to ensure a user’s data has been correctly inserted after signup.

Ans12:
To handle dynamic objects, UFT uses techniques like regular expressions or descriptive programming. For example, if buttons on a page change text based on user actions, you can identify them using patterns in their properties:

```
Browser("MyApp").Page("MyApp").WebButton("text:=Click.*").Click
```

Here, the `text:=Click.*` matches any button starting with "Click", allowing UFT to handle dynamic UI elements easily.

Ans13:
Parameterization allows you to replace hardcoded values with variable data, making tests reusable with multiple inputs. It’s useful for testing login pages with different usernames and passwords. In UFT, you can do this by using the Data Table. For example, you store different usernames in rows and use them in your script:

```
Browser("App").Page("Login").WebEdit("username").Set DataTable("Username",
dtGlobalSheet)
```

This helps in running the same test for different data inputs automatically.

Ans14:
To read input from an Excel sheet and use it in a login test, first link the Excel sheet to the Data Table. Then, in your test script:

```
Browser("App").Page("Login").WebEdit("username").Set DataTable("Username",
dtGlobalSheet)
Browser("App").Page("Login").WebEdit("password").Set DataTable("Password",
dtGlobalSheet)
Browser("App").Page("Login").WebButton("Login").Click
```

Add different username-password combinations in the Data Table, and UFT will run the script for each row.

Ans15:
There are three types of parameters in UFT:

- **Test Parameters** are used to pass values when running a test from ALM or using a test batch.
- **Action Parameters** allow passing data between different actions in a test.
- **Data Table Parameters** are used for data-driven tests and come from the Excel-like sheet in UFT.
  Each is useful for controlling data at different scopes of the test execution.

Ans16:
An action in UFT is a container for a series of steps that perform a task. Actions help organize test logic into smaller parts like "Login", "Search", or "Logout". Reusable

actions can be called multiple times from different tests. For example, you can create a “Login” action that enters a username and password and clicks login. This makes maintenance easier since changes need to be made only once.

Ans17:
A function library in UFT is a separate file (.vbs or .qfl) that contains reusable functions. You can create it using the Function Library Editor, then associate it with your test using File > Settings > Resources. Once added, functions from the library can be called in any action. This helps in writing modular, reusable, and clean code across multiple tests.

Ans18:
Function Library code:

```
Function AddNumbers(a, b)
   AddNumbers = a + b
End Function
```

Test script:

```
result = AddNumbers(5, 7)
MsgBox "Sum = " & result
```

This demonstrates how to call a reusable function from your test script and display the result.

Ans19:
Descriptive Programming in UFT lets you define objects directly in code without relying on the Object Repository. It's useful when objects change frequently or for dynamic apps. For example, to click a “Submit” button:

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

This method helps in more flexible and script-based object handling, especially for advanced automation scenarios.

Ans20:
Descriptive Programming syntax uses key-value pairs to describe object properties. Example:

```
Set linkObj = Description.Create()
linkObj("micclass").Value = "Link"
linkObj("text").Value = "Contact Us"
Browser("title:=.*").Page("title:=.*").Link(linkObj).Click
```

Here, UFT creates a description for a link with specific properties and uses it to interact with the object, avoiding dependence on Object Repository.

Ans21:
UFT handles dynamic objects in Descriptive Programming by using patterns or regular expressions in property values. For example, if a link's text changes based on user actions like "Order 123", "Order 124", you can use:

```
Browser("title:=.*").Page("title:=.*").Link("text:=Order \d+").Click
```

This clicks any link with the text "Order" followed by numbers. This approach is very useful when object properties keep changing at runtime.

Ans22:
Synchronization is important because it makes sure that UFT waits for the application to be ready before performing actions. Without it, UFT might try to interact with an object that hasn't loaded yet. UFT provides techniques like the Sync method, Wait, WaitProperty, and Exist statements. These help align the speed of the script with the application's response time and reduce test failures.

Ans23:
Here's a script that uses Sync and Wait:

```
Browser("title:=.*").Page("title:=.*").Sync
Wait(3) ' Waits for 3 seconds
Browser("title:=.*").Page("title:=.*").WebButton("name:=Login").Click
```

`Sync` waits until the browser finishes loading. `Wait` gives a manual pause, which is useful for slower systems. These ensure that UFT doesn't act too fast and fail on slow-loading pages.

Ans24:
For slow or dynamic applications, use `WaitProperty` or `Exist` with a timeout. For example:

```
Browser("title:=.*").Page("title:=.*").WebButton("name:=Submit").WaitProperty
"enabled", True, 10000
```

This waits up to 10 seconds for the Submit button to become enabled. This type of wait is smarter than fixed waits and reduces test flakiness by checking real-time conditions.

Ans25:
You can handle pop-ups or alerts in UFT using recovery scenarios or by scripting alert handling directly. Example:

```
If Dialog("text:=Alert").Exist Then
   Dialog("text:=Alert").WinButton("text:=OK").Click
End If
```

Recovery scenarios are set up in UFT to automatically detect and handle unexpected events like pop-ups, missing files, or application crashes.

Ans26:
UFT generates a detailed test result report after each run. It shows the status (pass/fail), step-by-step details, and screenshots (if enabled). You can view results through the Test Results Viewer, which gives you a breakdown of what happened at each step, including errors or warnings. This helps in debugging and reporting progress.

Ans27:
The "Test Results" tab shows the outcome of each test step with logs, errors, and screenshots. The "Run-Time Data Table" captures the data used during execution.

It's useful to verify if parameterized data was correctly read. You'd use both to debug — results to see what failed and data table to see if the test used correct input values.

Ans28:
You can generate a custom report using VBScript and FileSystemObject. Example:

```
Set fso = CreateObject("Scripting.FileSystemObject")
Set file = fso.CreateTextFile("C:\UFTReport.txt", True)
file.WriteLine "Test Step: Login"
file.WriteLine "Status: Pass"
file.WriteLine "Message: Login successful"
file.Close
```

This creates a simple text file with custom messages. You can extend this to log more steps and results during your test.