PassQueen
a.k.a dumpsqueen.com

UiPath Advanced RPA Developer v1.0 Exam (UiARD)

UiPath UiPath-ARDv1

Version Demo

Total Demo Questions: 42

Total Premium Questions: 424

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Advanced UI Automation	78
Topic 2, Advanced Data Manipulation	86
Topic 3, Advanced Selectors	33
Topic 4, Orchestrator	70
Topic 5, REFramework	72
Topic 6, Error Handling and Debugging	53
Topic 7, Mix Questions	32
Total	424

QUESTION NO: 1

Which of the following statements are true regarding project dependencies in UiPath Studio?

A. Dependencies can be installed and updated from the Manage Packages window.

B. A project can use activities supplied by installed dependency packages.

C. Publishing a project removes all dependency version information.

D. Dependencies can only be added after a project has been published.

ANSWER: A B

Explanation:

A dependency identifies an activity package or library package that a UiPath project requires in order to use its activities, types, and functionality. Dependencies are managed from the Manage Packages window, where developers can install packages from configured feeds and select package versions appropriate for the project.

Publishing a project packages its required dependency information with the project, allowing the Robot to resolve the required packages when the automation is executed. Updating a dependency should be performed carefully, because a newer package version can introduce behavioral or compatibility changes that require validation. See the UiPath Managing Packages documentation and the UiPath Automation Projects documentation.

QUESTION NO: 2

In which types of variable can you store text?

Options are :

A. Integer

B. String

C. Double

D. GenericValue

ANSWER: B D

Explanation:

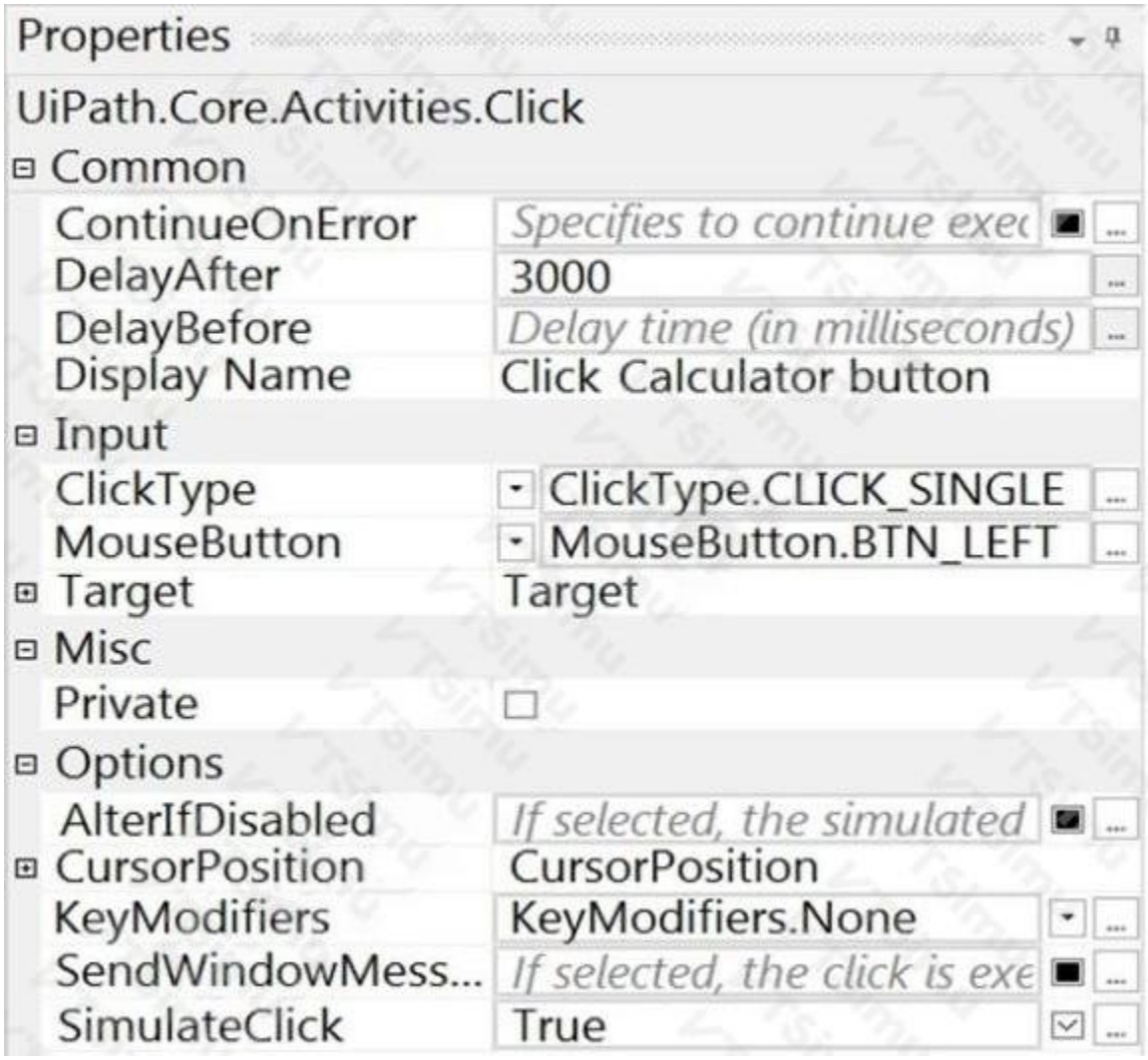
String is the standard UiPath/.NET variable type for textual data. It is intended for sequences of characters, including names, messages, file paths, URLs, email content, and values read from text-based user-interface elements. A String variable preserves the value as text and supports common text operations such as concatenation, comparison, replacement, splitting, and formatting.

GenericValue can also store text. UiPath uses this flexible variable type when a value may be represented as text, a number, a date, or another basic value during automation design. When assigned a textual value, GenericValue can hold that text and can be converted or used in expressions where an appropriate string representation is needed. This flexibility is useful when the source data type is not known in advance or may vary across executions, although String remains the clearest type when the workflow is specifically handling text.

UiPath's variable documentation describes variables as typed containers for storing and manipulating workflow data, while its data-type guidance identifies String and GenericValue as types applicable to text values. See UiPath Studio: About Variables and UiPath Studio: Managing Variables.

QUESTION NO: 3

A developer uses Workflow Analyzer with the default rules to check if a project follows best practices. In one of the workflows, the Properties of a Click activity is shown in the following exhibit.



Which Workflow Analyzer rule will trigger a warning for this activity?

A. Hardcoded Delays

B. Hardcoded Timeout

C. Activity Name Defaults

D. Simulate Click

ANSWER: D

Explanation:

Simulate Click is the rule that triggers the warning. The Click activity shown is configured without simulated input, so it uses the default hardware-based click behavior rather than the Simulate Click input method. UiPath Workflow Analyzer includes a rule that checks Click activities for this configuration because simulated clicks are generally preferred when the target application supports them. Simulated input is typically faster and more reliable for unattended automation because it does not require the target element to be physically available for a mouse action in the foreground. It can also reduce dependencies on screen focus and avoid issues caused by user interaction or changes in the active window.

The rule is a best-practice warning, not a guarantee that every Click activity must use simulated input. Some applications and controls do not support simulation, in which case another input mode may be necessary. In the displayed configuration, however, the absent simulated-click setting matches the condition evaluated by the Workflow Analyzer rule. UiPath documents Workflow Analyzer rules and their purpose in the Workflow Analyzer rules documentation. See also UiPath's Click activity documentation for input-method behavior.

QUESTION NO: 4

Where should credentials be stored? Select all the options that apply.

Options are :

A. In Windows Credential Store.

B. In Orchestrator, as assets.

C. Directly inside the workflows, as variables.

ANSWER: A B

Explanation:

Credentials should be kept in secure credential-management mechanisms rather than handled as ordinary process data. **In Windows Credential Store.** is appropriate when credentials are needed locally on a particular Windows machine or user profile. UiPath's credential activities can retrieve a stored username and password securely from Windows Credential Manager, allowing the automation to use the secret without embedding it in the process design.

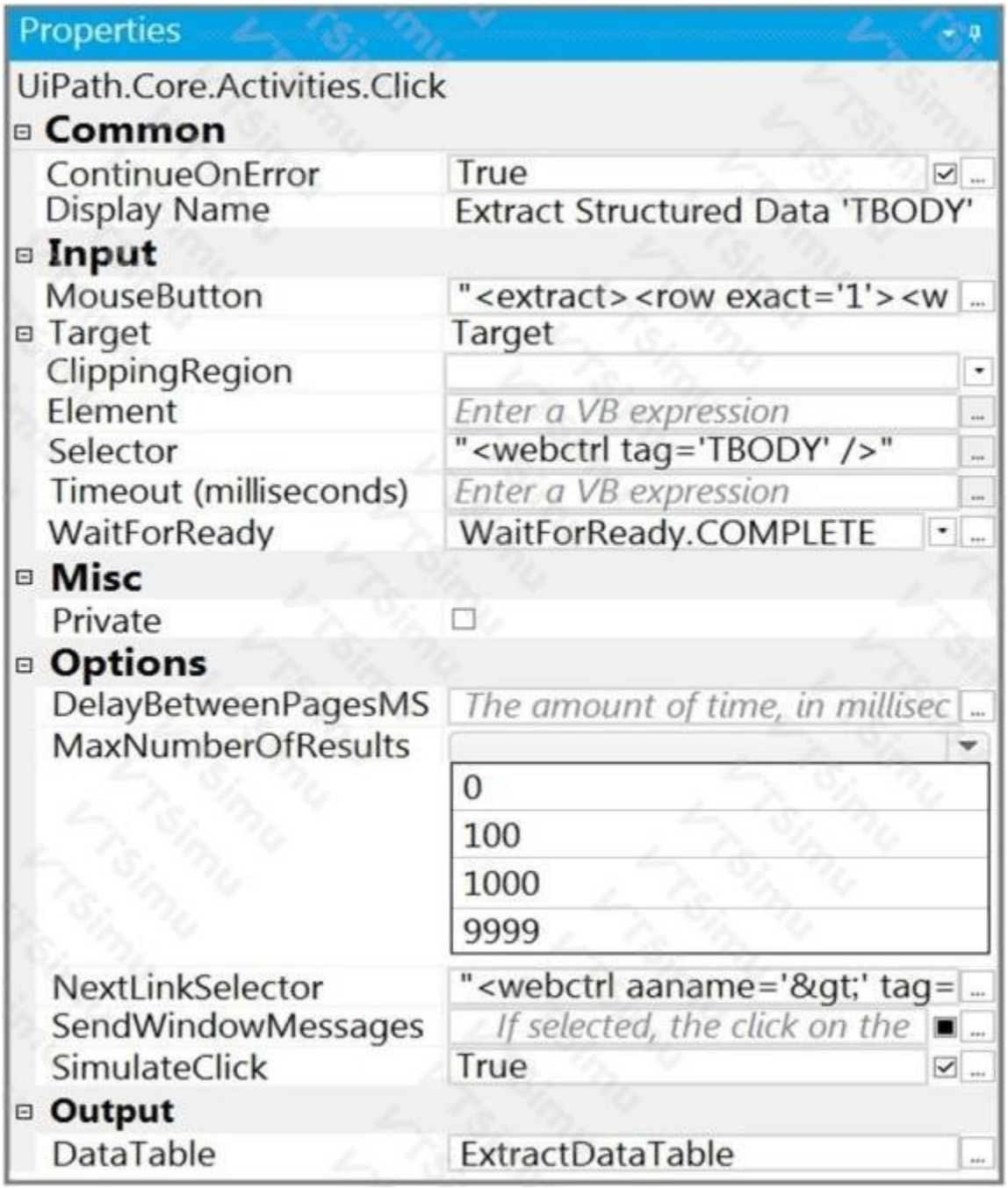
In Orchestrator, as assets. is also appropriate, specifically by creating a credential-type Asset. Credential Assets are centrally managed and can be assigned per robot or made available at the appropriate folder scope. When a process retrieves the asset at runtime, Orchestrator provides the credential values securely, supporting governance, reuse across automations, and credential rotation without editing the workflow itself. This is the preferred approach for unattended, shared, and enterprise automations because the credential is administered separately from the automation package.

UiPath documents credential assets and their secure handling in Managing credentials in Orchestrator. For local credential retrieval through UiPath activities, see Get Credential.

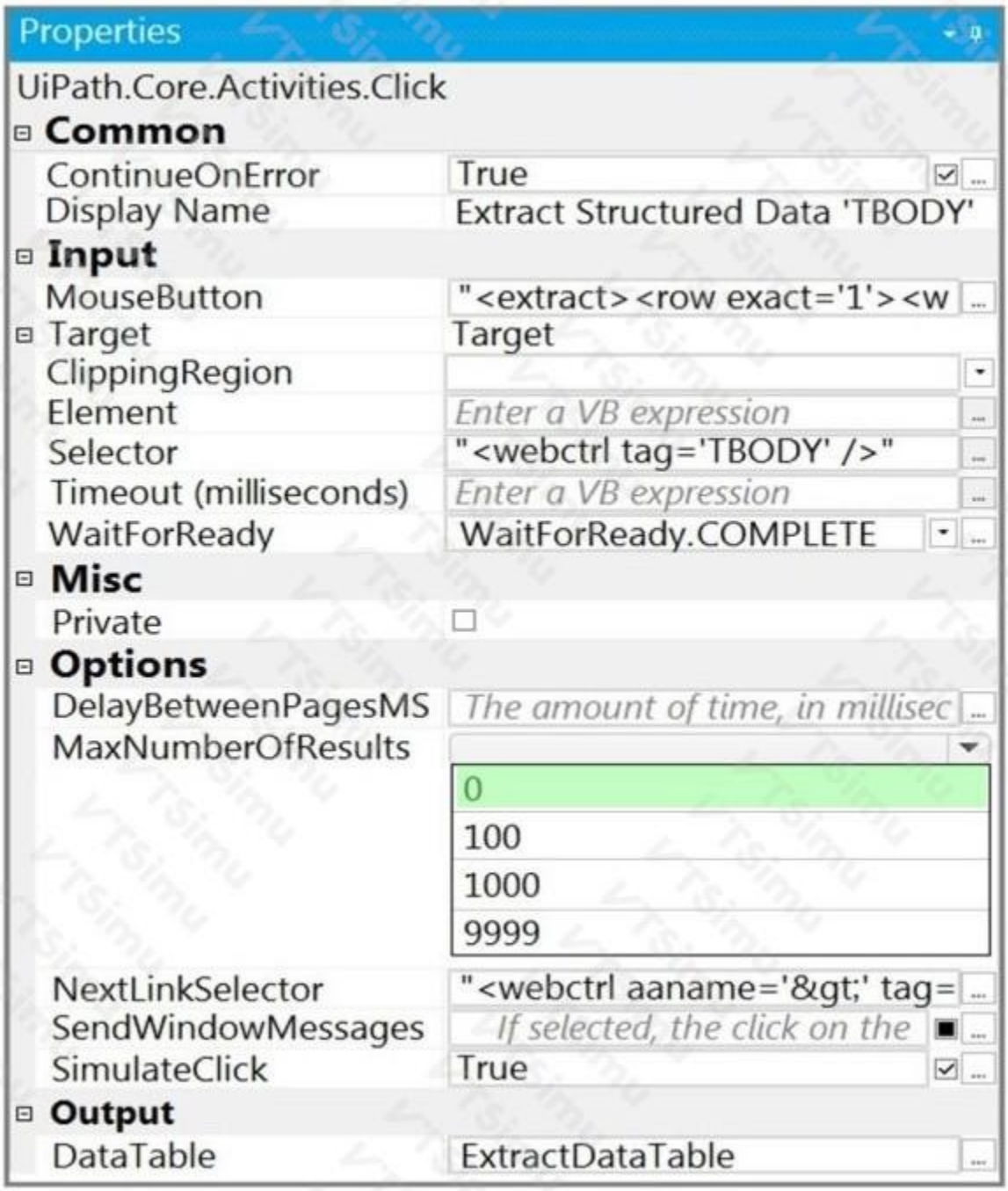
QUESTION NO: 5 - (HOTSPOT)

A developer has created a project to scrape structured data from Service Desk tickets. Upon review, the developer discovered that the Extract Structured Data activity was set to retrieve 50 results. The activity needs to be updated to retrieve all available results.

From the drop-down list shown in the exhibit, select the correct value that should be used to update the MaxNumberOfResults property.


Properties
UiPath.Core.Activities.Click
Common
ContinueOnError
True
Display Name
Extract Structured Data 'TBODY'
Input
MouseButton
"<extract><row exact='1'><w
Target
Target
ClippingRegion
Element
Enter a VB expression
Selector
"<webctrl tag='TBODY' />"
Timeout (milliseconds)
Enter a VB expression
WaitForReady
WaitForReady.COMPLETE
Misc
Private
Options
DelayBetweenPagesMS
The amount of time, in millisec
MaxNumberOfResults
0
100
1000
9999
NextLinkSelector
"<webctrl aaname='>' tag=
SendWindowMessages
If selected, the click on the
SimulateClick
True
Output
DataTable
ExtractDataTable

ANSWER:



Explanation:

The correct setting is **0**. In the Extract Structured Data activity, the **MaxNumberOfResults** property controls the maximum number of structured-data rows that the activity is allowed to return. A value of 0 has the special meaning of no result limit, so the extraction continues until all available matching ticket records have been collected. This is the appropriate configuration when the developer needs the DataTable output to contain the complete set of Service Desk ticket results rather than a fixed-size subset.

This property is particularly important for paginated tables. Once the structured-data wizard has identified the table and its next-page navigation, the activity can move through the available pages and accumulate the extracted rows. With the limit disabled through the value 0, the activity is not stopped merely because a predefined maximum count has been reached. The resulting DataTable can therefore be used for downstream processing, reporting, filtering, or writing the full ticket dataset to a file or database.

The green selection in the exhibit is placed on 0, which is exactly the value required to retrieve all available results. UiPath documents the Extract Structured Data activity and its configuration in the Extract Structured Data activity documentation. Using an unlimited result count is the correct choice whenever the automation's requirement is to scrape every available row returned by the target application.

QUESTION NO: 6

Which of the following statements are true regarding the Invoke Workflow File activity?

A. It can be used to execute a reusable XAML workflow file.

B. Arguments can be mapped between the calling and invoked workflows.

C. It can invoke only the Main.xaml file of another project.

D. Variables declared in the invoked workflow are automatically visible to the calling workflow.

ANSWER: A B

Explanation:

Invoke Workflow File runs another workflow file from the current automation. It supports building modular projects by separating reusable tasks, such as login, validation, reporting, or application cleanup, into focused workflow files.

Arguments are mapped on the activity so values can be passed into the invoked workflow and results can be returned to the calling workflow. The invoked workflow file must be available in the project at the configured path when the project is executed. See the UiPath Invoke Workflow File activity documentation and the UiPath workflow arguments documentation.

QUESTION NO: 7

A developer was assigned a task to build a process that will interact with hidden or minimized windows on an employee's machine.

To ensure the UI automation runs in the background which commonly used activity property must always be avoided?

A. Default

B. Simulate Type

C. Activate

D. SendWindowMessages

ANSWER: C

Explanation:

Activate is the property that must be avoided when a process is intended to interact with an application in the background. When enabled on a UI automation activity, Activate brings the target application window to the foreground and gives it focus before the activity performs its action. That behavior directly conflicts with the requirement to leave an employee's active desktop undisturbed while the automation works with a hidden or minimized target window.

For background-compatible UI automation, the developer should instead use an input approach that does not require foreground focus when the target technology supports it. UiPath's Simulate input mode is designed to send input through the application rather than relying on physical keyboard or mouse interaction, and window-message-based input can also support non-foreground operation in applicable applications. The key design principle is that no activity should be configured to explicitly activate the target window, because activation makes the automation visible and can interrupt the user's current work. UiPath documents activation as bringing a UI element or application to the foreground, while its UI automation guidance describes choosing appropriate input methods for reliable, non-intrusive interaction. See UiPath Activate activity documentation and UiPath UI Automation documentation.

QUESTION NO: 8

You need to collect employees data and send it by email as an Excel file. What type of workflow is the most suitable for the final part, which adds the file attachment, formats the email, and sends it?

Options are :

A. Sequence

B. Flowchart

C. State Machine

D. Directed Acyclic Graph (DAG)

ANSWER: A

Explanation:

Sequence is the most suitable workflow type because the described final stage is a simple, linear set of actions that should run in a defined order. The automation first prepares the email message and its formatting, then includes the generated Excel file as an attachment, and finally sends the completed message. A sequence presents activities consecutively from top to bottom, making this kind of short, ordered process easy to build, read, test, and maintain.

In UiPath Studio, sequences are intended for straightforward automation logic where activities follow one another without requiring complex branching or repeated transitions between distinct process states. The email-delivery portion can therefore contain the relevant mail activities and supporting assignments in one focused workflow, with the attachment path and message content supplied from the employee-data collection stage. Keeping this final delivery logic in a sequence also makes its execution order explicit: the email cannot be sent until its content and attachment have been prepared. UiPath's workflow design guidance describes sequences as suitable for linear processes, as documented in UiPath Studio: Sequences. For configuring the delivery activity itself, see UiPath Send Outlook Mail Message.

QUESTION NO: 9

How can you delay the Automatic Recording?

Options are :

A. By hitting the Escape key

B. By right clicking

C. Not possible

D. By hitting the F2 key

ANSWER: D

Explanation:

By hitting the F2 key is correct. In UiPath Studio's Automatic Recording workflow, pressing F2 activates a short delay before the recorder begins capturing actions. This gives the developer time to navigate to, open, or position the target application and its required screen state without having those preparatory actions included in the generated automation. After the delay, UiPath begins recording the user interactions and creates the corresponding activities, such as clicks, typing actions, and application interactions.

This feature is particularly useful when the action to automate is available only after opening a menu, dialog, or another transient interface element. Starting the recorder with a delay helps ensure that the recording begins at the intended point in the process, producing a cleaner sequence of generated activities and reducing subsequent editing. UiPath documents the automatic recording controls and notes that F2 provides the delayed-start behavior. See the official UiPath Studio recording documentation at UiPath Studio: Recording and the related Automatic Recording guide.

QUESTION NO: 10 - (SIMULATION)

In this exercise, you will create a UiPath automation that performs the steps below.

To achieve this, you will use the REFrameWork as the starting template and follow the UiPath development best practices.

Here are the steps performed by the Robot:

1. Log in to https://www.acme-test.com.

2. On the landing page, Dashboard, click or hover over the Vendors menu item and then click on Search for Vendor. Click on Display All Vendors. Scrape the data from the whole table isplayed. The resulting datatable will be used as the input data for the process. Navigate back to the dashboard.

Note: Navigation can be achieved in multiple ways by the robot - choose whichever you find best.

3. For each Tax ID:

- Navigate to Vendors - Search page (click or hover over the Vendors menu item and then click on Search for Vendor);

- Type the Tax ID into the Vendor Tax ID field;

- Click on Search;

- Extract the values for the Vendor, City and Country and compare them with the values from the previously extracted table from the Display All Vendors page (check for EXACT match for all fields!);

- If the values are not matching, this should be categorized as a Business Rule Exception;

- If the City does NOT belong to the group {""Paris"", ""Bucuresti"", ""Moscow"", ""Stuttgart"", ""Koln""}, this should be categorized as the second Business Rule Exception. We can only process requests from these cities. Check the City value extracted after the individual Tax ID search;

- If no Business Rule Exception, Append the resulting datatable from each page into an Excel worksheet;

you shouldn't worry about the headers and format of the output file.

Constraints to follow in the development, using the REFrameWork:

1. TransactionItem datatype should be a DataRow. The process should recover and retry 2 times in case of errors in navigation between the Vendor Search and Vendor Search Results pages. One transaction is the action of navigating to the Vendor Search page, searching for the TaxID and scraping the values from the resulting one row table. (Similar to ACME Process 5 from the UiPath Academy).

2. Create a separate workflow file for the Login to ACM

E.
File input arguments: URL ; Username ; Password .

3. Create a separate workflow file for closing ACM

E.

4. Add the ACME_URL and ACME_Credential to the Excel Config file.

5. Populate InitAllApplications.xaml from the Framework folder with Invoking the Login to ACME and navigation to the Work Items.

6. Populate CloseAllApplications.xaml from the Framework folder with Invoking the Close ACM

E.

7. Populate KillAllProcesses.xaml from the Framework folder with killing the process used.

8. Populate the Process.xaml file with the following actions: Navigation, Searching for TaxID, Scraping, Checking if the values match, Checking for the correct City, Appending to Excel.

Important Note: Don't use external file references outside of the project folder (including Orchestrator Assets). Place all the used files within the project folder, zip that folder and upload it to the UiPath Certification Platform.

Zip ALL the used workflow files AND the output Excel file. Then upload the .zip file to the UiPath Certification Platform.

ANSWER: See the explanation for the answer

Explanation:

Implement the project as a REFramework process with `DataRow` transaction items. The source transaction table is the complete Vendor table obtained from *Vendors → Search for Vendor → Display All Vendors*.

Add `ACME_URL` and `ACME_Credential` to `Data/Config.xlsx`. The URL value is `https://www.acme-test.com`; retrieve the credential through the configured local/project credential mechanism rather than using an external asset.
Set the framework retry setting to `2` (for example, `RetryNumber` in Config). This permits two retries when the transaction fails with a system exception during navigation/search/results scraping.
In `Main.xaml`, set `TransactionItem` / `out_TransactionItem` to type `System.Data.DataRow`.

Login.xaml should have these input arguments:

```
in_URL      As String
in_Username As String
in_Password As SecureString
```

It opens/attaches to ACME, navigates to `in_URL`, enters the username and password, and clicks Login.

CloseACME.xaml should close the ACME browser/application cleanly. Invoke `Login.xaml` from `Framework/InitAllApplications.xaml`, passing the URL from `Config("ACME_URL")` and the username/password obtained from `ACME_Credential`. Invoke `CloseACME.xaml` from `Framework/CloseAllApplications.xaml`. In `Framework/KillAllProcesses.xaml`, kill the browser process used for ACME (for example, `chrome`, `msedge`, or `firefox`, depending on the selected browser).

Initialization/source data:

1. After login, go to *Vendors → Search for Vendor*.
2. Click *Display All Vendors*.
3. Use Extract Table Data / Data Scraping to store the entire displayed table in a `DataTable`, for example `dtVendors`. It must include Tax ID, Vendor, City, and Country.
4. Navigate back to Dashboard. Pass/store `dtVendors` so that `GetTransactionData.xaml` can return one row at a time.

In `GetTransactionData.xaml`, return the row for the current transaction number, such as:

```
If in_TransactionNumber <= dtVendors.Rows.Count Then
    out_TransactionItem = dtVendors.Rows(in_TransactionNumber - 1)
Else
    out_TransactionItem = Nothing
End If
```

Process.xaml must process one `DataRow` as follows:

1. Read its Tax ID, e.g. `taxId = in_TransactionItem("Tax ID").ToString.Trim`.
2. Navigate to *Vendors → Search for Vendor*.
3. Enter `taxId` in *Vendor Tax ID* and click *Search*.
4. Scrape the single-row results table into a DataTable, for example `dtResult`.
5. Compare the searched row with the original transaction row using exact string equality for **Vendor, City, and Country**. A suitable check is:

   ```
   isMatch = dtResult.Rows.Count = 1 AndAlso

   dtResult.Rows(0)("Vendor").ToString.Equals(in_TransactionItem("Vendor").ToString,
   StringComparison.Ordinal) AndAlso

   dtResult.Rows(0)("City").ToString.Equals(in_TransactionItem("City").ToString,
   StringComparison.Ordinal) AndAlso

   dtResult.Rows(0)("Country").ToString.Equals(in_TransactionItem("Country").ToString,
   StringComparison.Ordinal)
   ```

6. If `isMatch` is false, use `Throw New BusinessRuleException("Vendor, City, or Country does not exactly match the Display All Vendors data.")`.

7. Check the City obtained from `dtResult`. The only allowed values are `Paris`, `Bucuresti`, `Moscow`, `Stuttgart`, and `Koln`. If it is not one of those values, throw the second business exception, for example:

   ```
   Throw New BusinessRuleException("City is not allowed for processing: " + city)
   ```

8. Only when both checks pass, append `dtResult` to an Excel worksheet inside the project folder, for example `Data/Output.xlsx`. Headers and formatting are not relevant.

Business Rule Exceptions are handled by REFramework as business exceptions and should not be retried. Unexpected selector, browser, navigation, search, or scrape failures must be allowed to propagate as system exceptions, so REFramework retries the transaction up to the configured two retries.

Submission: keep every workflow and the generated output workbook inside the project folder. Zip the complete project contents, including the Framework files, `Login.xaml`, `CloseACME.xaml`, Config workbook, and output Excel workbook; do not use external paths or Orchestrator Assets.

QUESTION NO: 11

Why is renaming activities considered to be one of the best practices?

Options are :

A. In case of an exception, to be able to find its source activity

B. To be able to understand the process logic without expanding each sequence or invoked workflow.

C. To easily understand the high-level business logic from a workflow.

ANSWER: A B C

Explanation:

Renaming activities with clear, business-meaningful names makes a UiPath automation substantially easier to build, review, troubleshoot, and maintain. The activity display name is visible directly in the Designer, so a descriptive name allows a developer or reviewer to understand the intent of a sequence, including a collapsed sequence or invoked workflow, without having to open every container and inspect its implementation. Meaningful names also provide a concise high-level view of the business steps performed by the workflow, such as validating input, retrieving a transaction, or posting an invoice.

Clear activity names are especially valuable during troubleshooting. When an execution error is reported, its workflow context and activity information help developers locate the failing step more quickly. Naming an activity for the action it performs gives that diagnostic information practical business context and reduces the time required to identify the source of an exception. UiPath's workflow-design guidance emphasizes creating readable, maintainable automations through clear organization and naming. See UiPath Workflow Design and UiPath Debugging.

QUESTION NO: 12

A developer automates a process which has an Excel file as input data; however, Orchestrator is not available. As a result, the developer needs to adapt the Robotic Enterprise (RE) Framework for use with tabular data.

Based on UiPath best practices, where should the Excel file be read and stored in a global DataTable variable?

A. In the new state in the Main.xaml that transitions from Init.

B. In the InitAllApplications.xaml workflow.

C. In the Get Transaction Data state in the Main.xaml.

D. In the Init state of the Main.xaml in the First Run sequence.

ANSWER: D

Explanation:

In the Init state of the Main.xaml in the First Run sequence is the appropriate place to load the Excel input into a DataTable whose scope is Main.xaml. REFramework separates one-time process initialization from the repeated transaction lifecycle. Reading the workbook during the first initialization run makes the complete input available before transaction processing starts, while avoiding unnecessary file reads for every transaction or initialization retry. The DataTable can then remain available to the transaction-acquisition logic, which uses the current transaction number or index to select the next row and create the transaction item. This approach is particularly suitable when Orchestrator queues are unavailable and the Excel worksheet is the local transaction source. It also keeps startup and input preparation centralized in the framework's initialization phase, while preserving the intended responsibility of the Get Transaction Data state: obtaining the next transaction from data that has already been prepared. UiPath's REFramework documentation describes the Init state as the stage for initial setup before transaction processing, and UiPath's Excel automation documentation covers reading workbook data into DataTables. See UiPath Robotic Enterprise Framework documentation and UiPath Excel activities documentation.

QUESTION NO: 13

What can the UiPath Robotic Enterprise Framework template be used as?

Options are :

A. A consumer of a queue in Orchestrator

B. A complete library for front office robots

C. The starting point for every automation project

ANSWER: C

Explanation:

The starting point for every automation project is correct because the Robotic Enterprise Framework (REFramework) is a reusable project template that provides a structured foundation for building dependable UiPath automations. It organizes an automation into clear stages, including initialization, retrieving and processing transactions, and ending the process. The template also includes established patterns for configuration management, logging, exception handling, retry behavior, and application recovery. Using this structure gives developers a consistent baseline that can be adapted to the process being automated, whether the work is transactional or requires process-specific changes. In practice, teams may choose a simpler template for a small task, but REFramework is designed as a broadly reusable starting architecture when a project benefits from production-ready control flow, resilience, and maintainability. It helps developers begin with common enterprise automation concerns already represented rather than designing those mechanisms from scratch. UiPath describes REFramework as a state-machine-based template intended for robust automation projects and documents its framework stages and components in the Robotic Enterprise Framework guide. The available project templates and their purpose are also described in the UiPath Studio project documentation.

QUESTION NO: 14

How should exceptions be handled? Select all the options that apply.

Options are :

A. By using Try Catch activities inside the workflow for unexpected application exceptions.

B. UiPath handles exceptions by default.

C. By validating data using conditional blocks for business exceptions.

ANSWER: A C

Explanation:

Using Try Catch activities inside a workflow is the appropriate way to handle unexpected application exceptions. A Try Catch activity lets the automation place error-prone operations in the Try section and define a controlled recovery path in Catch, such as logging diagnostic details, taking a screenshot, retrying when appropriate, cleaning up resources, or propagating the exception to a higher-level handler. This makes failures explicit and allows the process to end or recover predictably rather than failing without contextual handling.

Validating data with conditional blocks is appropriate for business exceptions because these situations arise when an application is functioning technically but the data or process state does not meet a defined business rule. For example, a required field may be missing, an invoice amount may exceed an approval threshold, or a transaction may already have been processed. Activities such as If or Switch can evaluate these expected conditions and route the workflow through the relevant business-exception path. In transaction-based automations, this approach can be combined with the framework's business-exception handling to record the outcome and continue with later transactions as designed. See UiPath's Try Catch activity documentation and exception-handling guidance.

QUESTION NO: 15

What can be used to debug a workflow?

Options are :

A. Breakpoints

B. The Slow Step option.

C. Highlighting activities.

ANSWER: A B C

Explanation:

Breakpoints, the Slow Step option, and highlighting activities are all UiPath Studio debugging capabilities. Breakpoints pause workflow execution immediately before a selected activity, allowing the developer to inspect the execution state, variables, arguments, and output in panels such as Locals and Watch. This makes them useful for isolating a specific point in a process where unexpected behavior occurs.

The Slow Step option runs activities with a delay between them, making the execution sequence easier to observe without manually pausing after every activity. It is especially useful when troubleshooting workflows that move too quickly to follow during normal debugging.

Highlighting activities provides visual feedback in the Designer about the activity currently being executed. This helps a developer follow the workflow path and identify where execution is at a given moment, particularly in larger or more deeply nested workflows. Used together, these capabilities support both controlled execution and clear visibility into what the automation is doing. UiPath documents breakpoint management and debug execution controls in its Debugging in Studio guide, and describes execution visualization features in the Debugging actions documentation.

QUESTION NO: 16

What direction can the arguments of a workflow have?

Options are :

A. In arguments

B. Out arguments

C. In/Out arguments

ANSWER: A B C

Explanation:

Workflow arguments in UiPath support all three listed directions: In, Out, and In/Out. “In arguments” pass values from the invoking workflow into the called workflow, allowing it to receive required inputs such as a file path, transaction item, or configuration value. “Out arguments” pass a value generated by the called workflow back to its caller, such as a calculated result or a status message. “In/Out arguments” both receive an initial value and return its updated value after the invoked workflow completes; this is useful when a workflow must modify a value supplied by its caller. These directions are set in the Arguments panel for a workflow and control how values are mapped in the Invoke Workflow File activity. UiPath’s workflow design model therefore provides a clear contract between reusable workflows and their callers, with arguments defining the data exchanged at invocation boundaries. See UiPath’s documentation on workflow arguments and Invoke Workflow File.

QUESTION NO: 17

A developer created automation in UiPath to process CVs of job candidates. This process is designed to help the HR team in their daily activities. Every day robot needs to process mails from the HR team sent to its Outlook account. Besides emails from HR, the robot also receives organizational emails and emails from other employees. There are several possible ways to set up the Get Outlook Mail Messages activity for the robot to only extract emails from the HR Team .

Please choose the incorrect way.

A. "[SenderName] = 'HR Team'" in the Filter property of the Get Outlook Mail Messages activity.

B. "[From] = 'HR Team'" in the Filter property of the Get Outlook Mail Messages activity.

C. "[SenderEmailAddress] = 'hr.team@company.com'" in the Filter property of the Get Outlook Mail Messages activity.

D. "HR Team" in the From property of the Get Outlook Mail Messages activity.

ANSWER: D

Explanation:

"HR Team" in the From property of the Get Outlook Mail Messages activity is the incorrect configuration because the classic UiPath *Get Outlook Mail Messages* activity does not expose a **From** input or property for selecting messages. The activity retrieves messages from the configured Outlook account and mail folder, and its built-in mechanism for restricting the returned messages is the **Filter** property. This filter is passed as an Outlook restriction expression, allowing the automation to request only messages whose sender-related fields match the HR mailbox or sender name before the workflow processes the results.

Using the activity’s Filter property is important in an HR CV-processing workflow because it narrows the collection at retrieval time, reducing the chance that newsletters, organization-wide announcements, or unrelated employee messages enter the candidate-processing logic. The workflow can then iterate only through the intended HR messages and process their attachments or content consistently. UiPath documents the available configuration of the activity, including its Filter field, in the Get Outlook Mail Messages documentation. The filtering expression follows Outlook’s item-restriction model, described in Microsoft’s Items.Restrict documentation.

QUESTION NO: 18

Which of the following methods can improve the reliability of a dynamic selector?

A. Replace a changing attribute value with a wildcard.

B. Use stable attributes that uniquely identify the target.

C. Include every available attribute in the selector.

D. Use an idx attribute whenever it is available.

ANSWER: A B

Explanation:

Replacing a changing attribute value with a wildcard makes a selector more resilient when part of an attribute changes between executions. For example, a changing title, identifier, or generated value can be matched using a wildcard while stable attributes continue to identify the intended element.

Using stable attributes that uniquely identify the target is also recommended. A selector should contain enough reliable information to distinguish the intended element without depending on volatile values such as changing indexes or dynamic identifiers. UiPath provides selector-validation and editing tools to support this work. See the UiPath selectors documentation and the UI Automation activities documentation.

QUESTION NO: 19

Which of the following statements are true about the Finally section in a Try Catch activity?

A. It is executed whether or not an exception occurs in the Try section.

B. It can be used to close applications and release resources.

C. It is executed only when no Catch block handles the exception.

D. It can contain only Log Message activities.

ANSWER: A B

Explanation:

The Finally section is executed after the Try section and any applicable Catch section, whether or not an exception occurred. It is therefore appropriate for cleanup actions that should be attempted regardless of the result of the protected workflow logic.

Closing an application and disconnecting a database connection are typical examples of cleanup operations placed in Finally. The Finally section does not replace exception handling: Catch blocks are still used when the workflow must respond differently to particular exception types. See UiPath documentation for Try Catch and exception handling.

QUESTION NO: 20

Which of these are workflow types available in UiPath Studio; Options are :

A. REFramework.

B. Flowchart.

C. Sequence.

D. Activity.

ANSWER: B C

Explanation:

Flowchart and **Sequence** are workflow types that can be created in UiPath Studio. A Sequence organizes activities in a linear, ordered flow and is well suited to straightforward automations whose steps generally execute one after another. It can also contain nested activities and other workflows, making it a common default design choice for smaller, focused processes.

A Flowchart represents automation logic as connected nodes and transitions. This design is particularly useful where the process contains branching, decisions, loops, or several possible routes between steps, because the visual layout makes the paths and their relationships easy to understand. UiPath Studio also supports State Machine workflows, although that workflow type is not included among the supplied choices. The UiPath workflow-design guidance describes Sequences, Flowcharts, and State Machines as the principal workflow forms and explains when to use each. See UiPath Workflow Types documentation and UiPath About Workflows documentation.

QUESTION NO: 21

How should a UiPath developer handle frequent changes in the project files?

Options are :

A. Old versions of the project files are not relevant

B. By creating daily backups of the files

C. By using a source control solution, such as SVN, TFS, etc.

ANSWER: C

Explanation:

By using a source control solution, such as SVN, TFS, etc. is the appropriate approach because source control provides a structured, traceable way to manage ongoing changes to UiPath automation projects. A repository records the history of changes, identifies who made each change, and supports comparison or restoration of prior versions when a regression or unintended update occurs. It also enables several developers to work on the same project in a controlled manner through commits, branches, merging, and conflict resolution. This is particularly valuable for enterprise automations, where changes must be reviewed, tested, and auditable before deployment. UiPath Studio supports integration with source-control systems, allowing developers to connect a project to a repository and use source-control operations directly while developing. UiPath's project-development guidance recommends version control as part of a collaborative development workflow, helping teams maintain a reliable project history rather than relying on manually maintained file copies. See UiPath's Git integration documentation and its TFS integration documentation.

QUESTION NO: 22

Which of the following types of variables can be defined in UiPath Studio?

Options are :

A. DataTable

B. GenericValue.

C. Number.

ANSWER: A B

Explanation:

DataTable and **GenericValue.** can both be defined as variable types in UiPath Studio. A DataTable variable holds structured, in-memory tabular data made up of rows and columns. It is commonly used with activities that read, filter, iterate through, transform, or write spreadsheet and database-style data. For example, the output of an Excel or database read operation can be stored in a DataTable variable and then processed using row-based activities or expressions.

GenericValue is a UiPath type intended to hold values whose exact type may vary during automation execution. It supports convenient conversion between common value forms, such as text, numeric values, Boolean values, and dates, when an activity or workflow needs flexible handling of a value. This type was especially useful in classic UiPath projects and remains a valid variable type available through UiPath's type selection mechanisms.

Variables provide named storage locations whose type determines the values and operations available in an automation. UiPath Studio lets developers create variables from the Variables panel and select an appropriate .NET or UiPath-supported type for the workflow's data requirements. See UiPath's documentation on variables and managing variables.

QUESTION NO: 23

Which statement about the UiPath Robotic Enterprise Framework template is false?

Options are :

A. The framework can be used only if you get the input data from the UiPath server queues.

B. The framework has a robust exception handling scheme and event logging.

C. The framework is meant to be a template that helps the user design processes.

ANSWER: A

Explanation:

The statement “The framework can be used only if you get the input data from the UiPath server queues.” is false because Robotic Enterprise Framework (REFramework) is a reusable state-machine project template, not a queue-only automation model. Its default transaction-processing design supports retrieving work items from UiPath Orchestrator queues, but developers can adapt the initialization and transaction-data logic for other sources. For example, a process can build a DataTable from an Excel file, database query, API response, or another application, then use each row or item as a transaction. The template’s purpose is to provide a dependable project structure for enterprise automations, including initialization, transaction processing, end-of-process handling, configurable settings, logging, and recovery patterns. Queue-based input is a common and especially useful implementation because queues provide centralized workload management and transaction status tracking, but it is not a prerequisite for using the framework. UiPath’s documentation describes REFramework as a template intended for transactional business processes and explains that its framework logic can be configured for the process being automated. See UiPath’s Robotic Enterprise Framework documentation and UiPath Orchestrator Queues documentation.

QUESTION NO: 24

A developer created an automation project in the Robotic Enterprise (RE) Framework which needs to enter a User ID depending on the machine it runs on. The User ID is stored in Orchestrator as a Text asset using the Value Per Account-Machine method.

Which steps should the developer perform to use this asset in the project?

A. Use the Get Asset activity in the workflow to get the User ID.

B. In the workflow, retrieve the User ID by referencing the Config dictionary.

C. Use the Get Asset activity in the workflow to get the User ID.

D. In the workflow, retrieve the User ID by referencing the Config dictionary.

ANSWER: B

Explanation:

In the REFramework, the appropriate approach is to configure the Orchestrator asset in the `Assets` sheet of `Config.xlsx` and then retrieve its value from the `Config` dictionary in the workflow. During initialization, REFramework's `InitAllSettings.xaml` reads the configured asset entries and obtains their values from Orchestrator, making them available through `in_Config`. For example, a workflow can assign the User ID using `in_Config("UserID").ToString`, where `UserID` is the key configured for that asset. Because the asset uses Value Per Account-Machine, Orchestrator resolves the value associated with the account and machine context of the robot that executes the job. This preserves REFramework's centralized configuration pattern while allowing the same deployed process to use the correct machine-specific User ID without hard-coded values or workflow-specific asset retrieval. UiPath documents the REFramework configuration-file structure, including its asset configuration, in the Robotic Enterprise Framework guide. The account-machine scoping behavior is described in the Orchestrator assets documentation.

QUESTION NO: 25

In a UiPath project, a developer uses a Click Button activity. A desktop application should be opened as a result of clicking the button. Based on best practice, what should the developer use to ensure: (1) the button element is clicked and (2) the application is opened correctly?

A. Modify the Click activity used to open the application by setting the ContinueOnError property to “True”

B. Use an Element Exists activity after the click to ensure that an element exists within the opened application

C. Use the Click activity within the Try Block and in the Catch Block to open the application

D. Use the Click activity in the Action section of a Retry Scope activity with a Condition that an element exists within the opened application

ANSWER: D

Explanation:

Use the Click activity in the Action section of a Retry Scope activity with a Condition that an element exists within the opened application. This pattern verifies the actual business outcome of the UI interaction rather than merely attempting the click. The Action section performs the Click activity, while the Condition checks for a reliable UI element that is unique to, and available only after, the target desktop application has opened. If that element is not detected, Retry Scope runs the action again until the condition succeeds or its configured retry limits are reached. This is especially useful in desktop automation because an application may take variable time to launch, a click may initially be missed due to temporary UI responsiveness issues, or the resulting window may not be ready immediately. Choose a stable target element in the opened application, such as a main window control or a distinct title-area element, and configure appropriate retry interval and timeout values for the application’s expected startup time. UiPath documents that Retry Scope repeats an action until a condition is met, making it suitable for this click-and-verify scenario. See the UiPath Retry Scope documentation and the UiPath Element Exists documentation.

QUESTION NO: 26

Which of the following statements are true regarding the OCR screen scraping method?

A. It can retrieve text that is displayed as an image.

B. Its accuracy can be affected by image quality and screen resolution.

C. It requires the target application to expose native UI text attributes.

D. It always retrieves the font size and color of each recognized character.

ANSWER: A B

Explanation:

OCR is useful when text is available only as an image, such as in remote desktop technologies, virtualized applications, scanned documents, or other interfaces where normal UI text is not exposed. It recognizes characters visually and returns the recognized result as text.

Because OCR depends on the visual appearance of characters, its accuracy can be affected by font quality, resolution, contrast, scaling, and image clarity. It does not provide reliable font formatting information merely by extracting text. UiPath provides multiple OCR engines and configuration options to support different automation scenarios. See the UiPath Screen Scraping documentation and the UiPath OCR activities documentation.

QUESTION NO: 27

Which of the following statements related to Orchestrator are true?

Options are :

A. A robot can execute many different jobs at the same time.

B. Robots can be assigned to multiple environments.

C. A robot can execute many different jobs one after the other.

ANSWER: B C

Explanation:

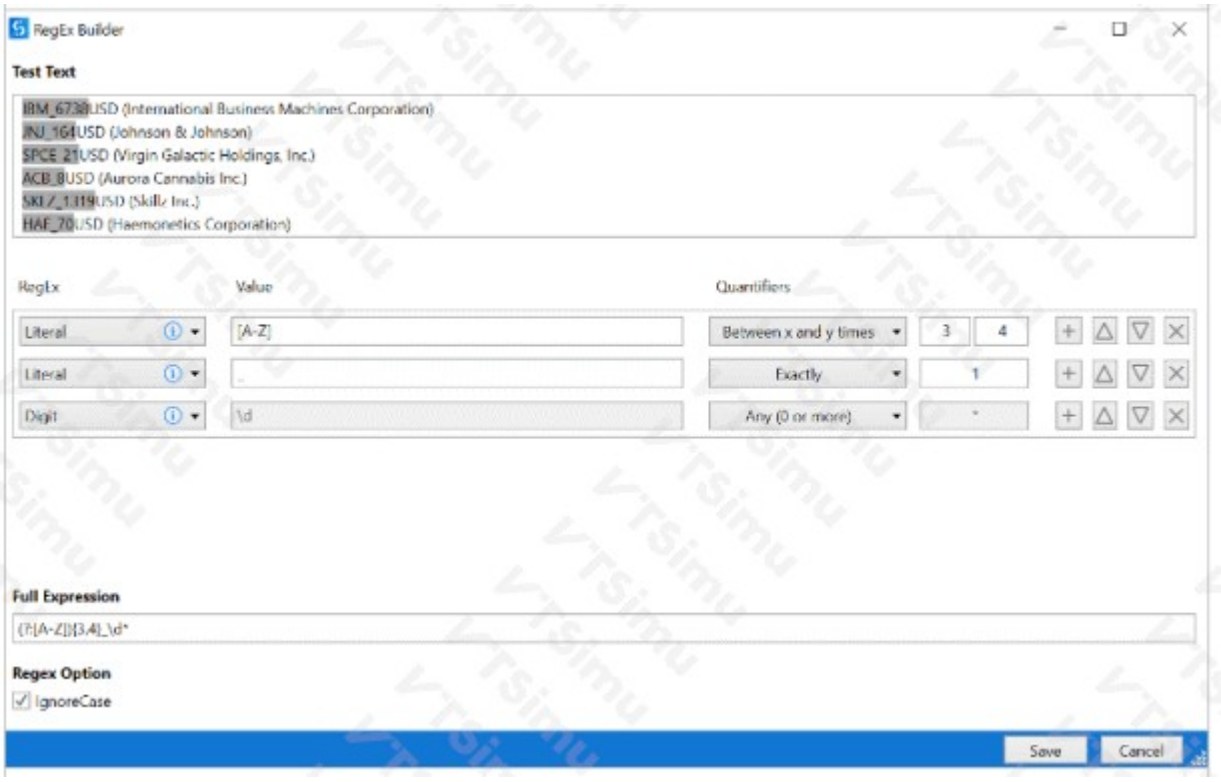
In the classic Orchestrator model used by this exam, robots are associated with environments, and the same robot may be assigned to more than one environment. This lets an organization make a robot available for different groups of processes or scheduling contexts without creating a separate robot definition for every environment. When a job is started from an applicable environment, Orchestrator sends the process to the selected robot according to its availability and configuration.

A robot can also execute different jobs sequentially. A job is an execution of a published process, and after the robot completes one job and returns to an available state, Orchestrator can use that robot to run another job. This is a fundamental scheduling pattern: multiple process executions can be queued or scheduled over time and handled by the same robot, one execution after another. The number of simultaneous executions is governed by the robot's available runtime capacity, whereas sequential execution naturally reuses the robot after each job finishes. UiPath's Orchestrator documentation describes robots and their execution capacity, while the job documentation explains that jobs represent process executions: UiPath: About Robots and UiPath: Managing Jobs.

QUESTION NO: 28

A developer has created a process that gathers a listing of stock market prices in the following format _USD ().

How should the third line in RegEx Builder be modified to ensure that only items with prices of at least 100 are identified?



A. RegEx: One of
Value: 123456789
Quantifiers: Between 2 and 4

B. RegEx: Any word character
Value: \w
Quantifiers: Exactly 3

C. RegEx: Digit
Value: \d
Quantifiers: Exactly 3

ANSWER: C

Explanation:

RegEx: Digit
Value: \d
Quantifiers: Exactly 3 is the appropriate RegEx Builder configuration for the expected three-digit price portion. In regular expressions, `\d` represents one decimal digit. Applying the *Exactly 3* quantifier creates the pattern `\d{3}`, which requires three consecutive numeric characters at that point in the stock-price format. When that numeric group is positioned directly before the literal `USD` text, it identifies price values from 100 through 999, so a one- or two-digit price cannot satisfy the required pattern. This is a clear way to constrain the numeric component without matching letters, underscores, or the surrounding company-name content. UiPath's RegEx Builder exposes common regex tokens and quantifiers visually, then generates the corresponding regular-expression syntax for use in matching activities. The definition of `\d` and the exact-count quantifier behavior are documented in the .NET decimal digit character class documentation and the .NET quantifiers documentation.

QUESTION NO: 29

Where should Credentials be stored?

Options are :

A. In Windows Credential Store

B. In Orchestrator as assets

C. Directly inside the workflows as variables

ANSWER: A B

Explanation:

Credentials should be stored in secure, purpose-built credential repositories rather than exposed in automation logic. **In Windows Credential Store** is correct because UiPath can use Windows Credential Manager to securely save and retrieve credentials on the machine or user profile where the automation runs. UiPath activities can retrieve the credential when needed, allowing the workflow to work with a secure credential object instead of a hard-coded username and password. UiPath documents this credential-management approach in its credentials documentation.

In Orchestrator as assets is also correct because a Credential asset is designed for centrally managed, encrypted username-and-password pairs. Orchestrator enables credentials to be assigned at the appropriate folder scope, consumed by authorized robots and processes, and maintained independently of the workflow package. This supports secure deployment across multiple environments and makes credential rotation possible without editing or republishing the automation. UiPath's Orchestrator Assets documentation describes Credential assets as secure assets used to store usernames and passwords. Using either approved store keeps sensitive values out of workflow source and supports safer, maintainable unattended automation.

QUESTION NO: 30

How can you improve a selector?

Options are :

A. By using intermediate containers for a better matching of the UI element.

B. By adding the absolute position of the elements to the selector.

C. By replacing the dynamic parts of an attribute with wildcards.

ANSWER: A C

Explanation:

Selectors are more reliable when they identify a target through stable, meaningful UI attributes and enough context to distinguish it from similar elements. “By using intermediate containers for a better matching of the UI element.” is correct because including a stable parent or container can narrow the search scope and uniquely identify the intended target, particularly when several controls have similar attributes. The selected containers should themselves be dependable and should not introduce volatile properties.

“By replacing the dynamic parts of an attribute with wildcards.” is also correct. UiPath supports wildcard characters in selector attribute values, allowing a selector to tolerate portions of an identifier or title that vary between executions while retaining the stable portion that identifies the application element. This is particularly useful for attributes containing generated IDs, changing document names, or session-specific values. Use wildcards deliberately and retain sufficient stable attributes or hierarchy so the resulting selector remains specific enough to match only the intended element. UiPath’s selector guidance discusses using wildcards for dynamically changing attributes and using UI hierarchy to construct robust selectors. See UiPath Selectors documentation and UiPath Wildcards in Selectors documentation.

QUESTION NO: 31

In the UiPath Robotic Enterprise Framework template, in the Main workflow, the State Machine includes the following states:

Options are :

A. Init State

B. Get transaction data State

C. Process Transaction State

D. Set Transaction State

E. End Process State

ANSWER: A B C E

Explanation:

The Robotic Enterprise Framework (REFramework) Main workflow is implemented as a state machine that organizes an unattended transactional automation into its standard lifecycle. **Init State** performs startup work, including initializing applications and system resources through the framework’s initialization workflow. **Get transaction data State** obtains the next transaction item from the configured transaction source and determines whether additional work remains. **Process Transaction State** runs the business-specific processing for the current transaction and supports the framework’s transaction-oriented exception handling and retry behavior. **End Process State** is the controlled shutdown stage, where REFramework closes applications and performs cleanup through its end-process workflow.

These states provide the framework’s core sequence: initialize once, repeatedly retrieve and process transaction items, then end the automation cleanly once no work remains or the process is stopped. This state-machine structure is a key reason REFramework is well suited to robust, queue-based or transaction-based automations. UiPath describes REFramework as a project template designed for transactional business processes and documents its workflow structure in the Robotic Enterprise Framework documentation. Further implementation guidance is available in UiPath’s REFramework Studio documentation.

QUESTION NO: 32

A developer created an automation project in the Robotic Enterprise (RE) Framework which needs to enter a User ID depending on the machine it runs on. The User ID is stored in Orchestrator as a Text asset using the Value Per Account-Machine method.

Which steps should the developer perform to use this asset in the project?

A. Add a row in the Settings sheet in Config.xlsx with the name of the Orchestrator asset.
Use the Get Asset activity in the workflow to get the User ID.

B. Add a row in the Settings sheet in Config.xlsx with the name of the Orchestrator asset.
In the workflow, retrieve the User ID by referencing the Config dictionary.

C. Add a row in the Assets sheet in Config.xlsx with the name of the Orchestrator asset.
Use the Get Asset activity in the workflow to get the User ID.

D. Add a row in the Assets sheet in Config.xlsx with the name of the Orchestrator asset.
In the workflow, retrieve the User ID by referencing the Config dictionary.

ANSWER: D

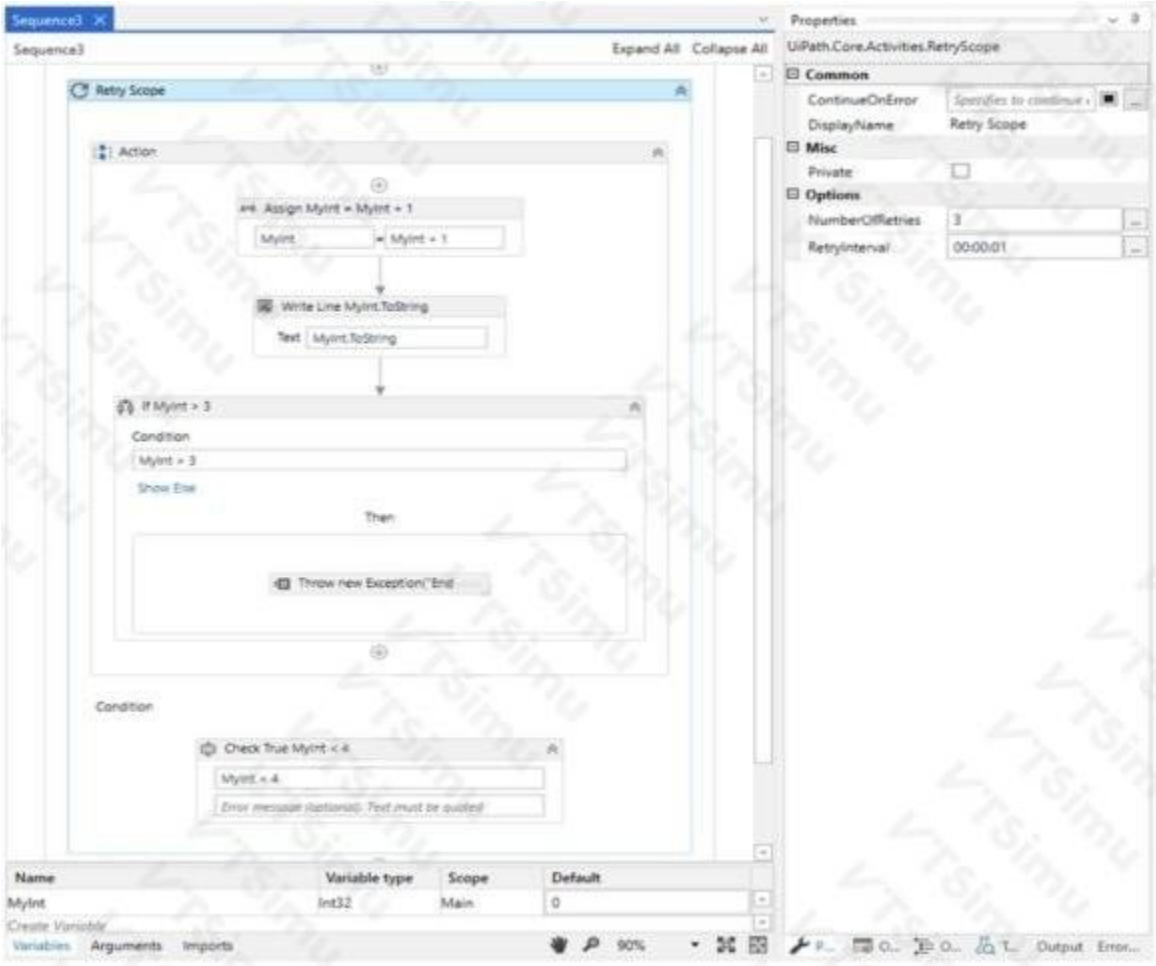
Explanation:

“Add a row in the Assets sheet in Config.xlsx with the name of the Orchestrator asset. In the workflow, retrieve the User ID by referencing the Config dictionary.” is correct because the REFramework is designed to centralize process settings and Orchestrator assets through `Config.xlsx`. The asset must be declared on the **Assets** sheet, rather than the Settings sheet, so that the framework’s initialization logic recognizes it as an Orchestrator asset. During initialization, REFramework retrieves the configured asset value and stores it in the shared `Config` dictionary. Subsequent workflows can therefore use the resolved User ID through the dictionary, for example with an expression such as `Config("UserID").ToString`.

The Value Per Account-Machine setting is resolved by Orchestrator according to the account and machine context under which the unattended process executes. This makes the same automation project portable across machines while allowing each applicable account-machine combination to receive its appropriate User ID. Using the framework’s asset configuration and shared dictionary also keeps the business workflows independent of direct asset-retrieval implementation details. UiPath documents asset retrieval in the Get Asset activity documentation and describes REFramework configuration in the Robotic Enterprise Framework documentation.

QUESTION NO: 33

A developer created a sequence with a Retry Scope shown in the following exhibit:



What is the content of the Output panel after running this sequence?

A. 0

B. 1

C. 1
2
3

D. 1
2
3
Retry Scope: End

ANSWER: B

Explanation:

1 is the content written to the Output panel. A Retry Scope runs the activities in its Action section and then evaluates the activities in its Condition section. When the condition is satisfied on the initial evaluation, the scope completes successfully rather than beginning another retry cycle. In the shown sequence, the logging activity writes the current value, which is 1, during that first Action execution. The condition succeeds at that point, so the Retry Scope has no reason to repeat the Action activities. Consequently, there is only one log entry, containing 1, in the Output panel. This behavior is important when designing retry logic: retries are not performed a fixed number of times merely because a retry count is configured. The configured retry count sets the maximum number of attempts available if the condition does not succeed; it does not require every attempt to occur. UiPath documents that Retry Scope repeatedly executes its action until its condition succeeds or the specified retry limit is reached. See the UiPath Retry Scope activity documentation and the UiPath control-flow activities documentation.

QUESTION NO: 34

What robot state is displayed on the Robots page while a job is being executed?

Options are :

A. Runnning

B. Pending

C. Busy

ANSWER: C

Explanation:

Busy is the robot state shown while the robot is actively executing a job. In UiPath Orchestrator, robot status reflects its current availability and activity. When Orchestrator assigns a job to a connected robot and execution has started, that robot is occupied by the running process and is therefore reported as Busy on the Robots page. This status is important operationally because it indicates that the robot is not currently free to accept another job; its execution capacity is being used until the current job completes, stops, faults, or otherwise releases the robot. Monitoring this state helps administrators and developers confirm that a dispatched process is actually being handled by a robot, distinguish active workload from available capacity, and plan unattended automation resources effectively. UiPath documents robot monitoring and the statuses displayed in Orchestrator in its Monitoring Robots guide. For the relationship between jobs, processes, and robot execution, see UiPath's Managing Jobs documentation.

QUESTION NO: 35

A developer is using UiExplorer to modify selectors. The "Highlight" button is present in UiExplorer. What is its functionality and when does this button appear?

A. Highlights the selected element from the Visual Tree in real time. The highlight stays on for 3 seconds.

B. Brings the target element in the foreground. The button is enabled only if the selector is valid.

C. Highlights the selected element from the Visual Tree in real time. The highlight stays on until it's switched off.

D. Brings the target element in the foreground. The button is enabled only if the selector is invalid.

ANSWER: B

Explanation:

Brings the target element in the foreground. The button is enabled only if the selector is valid. In UiExplorer, the general Highlight command is used to visually identify the UI element currently described by the selector. When activated, it brings attention to the matching target in the application, allowing the developer to confirm that the selector identifies the intended control before using it in an automation. This visual verification is particularly useful while refining attributes, handling dynamic values, or checking selector reliability.

The command is available only when UiExplorer can validate the selector. That validity requirement matters because UiExplorer must be able to resolve the selector to a target before it can show the corresponding element. After activation, the highlight remains active until the developer turns it off, rather than disappearing after a short timed interval. UiPath distinguishes this general selector-validation Highlight command from highlighting available in the Visual Tree, which is a separate contextual capability for visually tracking a selected tree item. See UiPath's UI Explorer documentation and its guidance on selectors.

QUESTION NO: 36

A developer automated a Performer process using the Robotic Enterprise (RE) Framework. Which state executes after the Process Transaction state has a result of "Success"?

A. Process Transaction

B. Get Transaction Data

C. End Process

D. Initialization

ANSWER: B

Explanation:

Get Transaction Data executes after a transaction is processed successfully in the standard REFramework flow. The Process Transaction state handles the business logic for the current transaction item and records its final outcome. Once that outcome is Success, the framework completes the current transaction cycle and transitions to Get Transaction Data to request the next item from the configured transaction source. This repeated cycle allows a Performer process to handle queue items, data-table rows, or other transaction sources one item at a time until no further data is available. The transition is part of the default state-machine design supplied by UiPath's REFramework template, which separates retrieving work from processing work and maintains transaction-level status and logging consistently. The framework's reusable implementation is available in UiPath's official REFramework repository. UiPath's State Machines documentation also describes how a workflow moves between states according to transition conditions; here, the successful result is the condition that directs execution back to obtaining the next transaction.

QUESTION NO: 37

Which of the statuses below can a transaction have? Select all the options that apply.

Options are :

A. New

B. Pending

C. In progress

D. Successful

E. Abandoned

F. Failed

G. Retried

H. Deleted

ANSWER: A C D E F G H

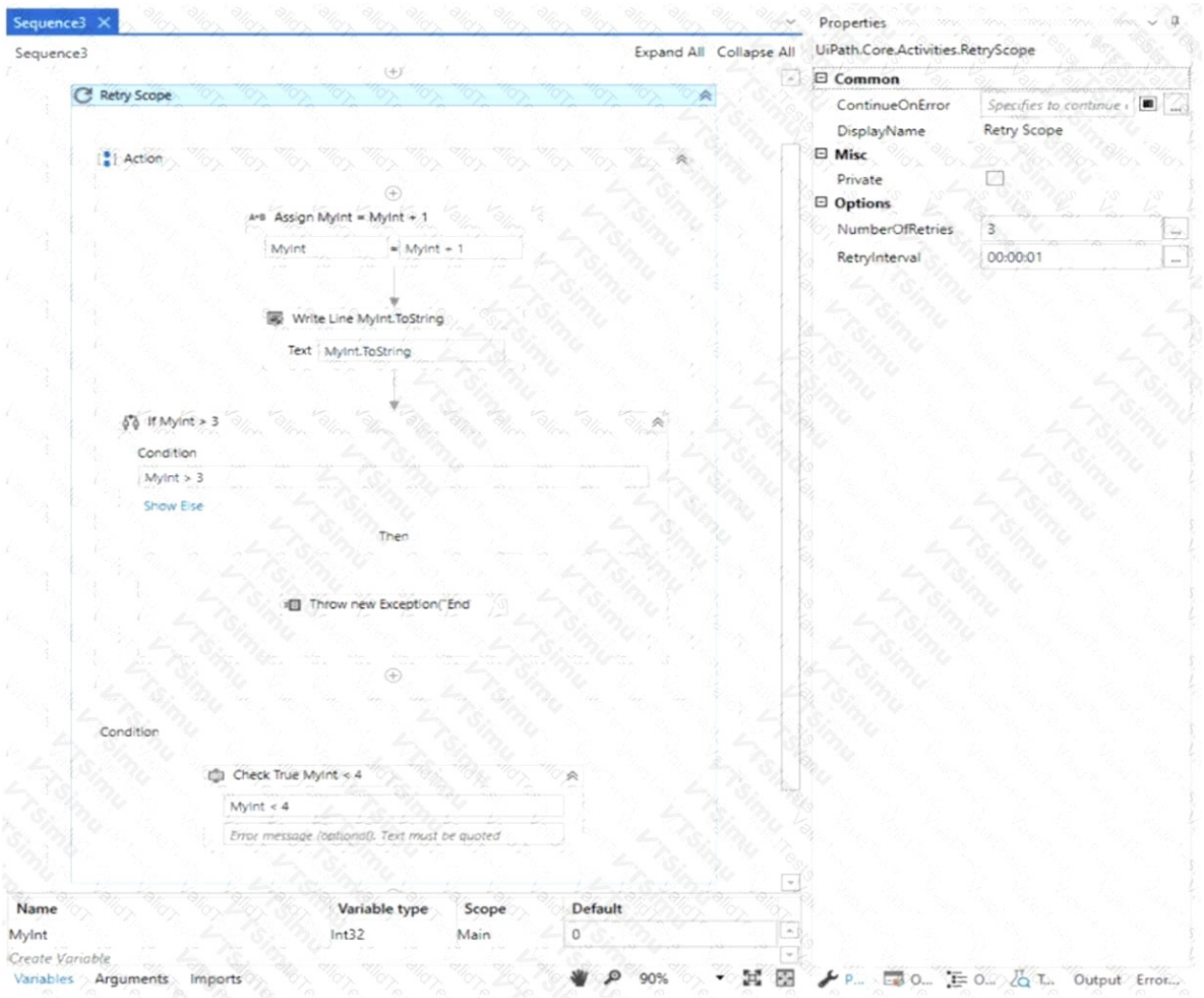
Explanation:

Orchestrator queue transactions use a defined lifecycle, and the applicable statuses are **New**, **In progress**, **Successful**, **Abandoned**, **Failed**, **Retried**, and **Deleted**. A newly created queue item starts as New. When a robot retrieves it for processing, its status becomes In progress. Once processing is reported as completed, Orchestrator records the result as Successful or Failed. If an item remains in progress beyond the queue's configured abandonment threshold, Orchestrator can classify it as Abandoned. Retry behavior is also tracked: when a queue item is retried according to the queue's retry settings, the original transaction is represented with the Retried status while a new attempt can be created for processing. Deleted is an available transaction status for queue items that have been deleted. These statuses enable queue monitoring, filtering, reporting, and operational handling of each transaction throughout its lifecycle. UiPath documents the queue-item

status model and queue-management behavior in its Orchestrator documentation: Queue item statuses and Managing queues.

QUESTION NO: 38

A developer created a sequence with a Retry Scope shown in the following exhibit:



What is the content of the Output panel after running this sequence?

A. 0

B. 1

C. 2

D. 2

ANSWER: B

Explanation:

`1` is the content written to the Output panel. In the illustrated workflow, the Retry Scope executes its Action section and then evaluates the Condition section after that action completes. The value displayed by the Write Line activity is therefore the value produced during the relevant Action execution shown in the exhibit. Once the condition evaluates successfully, the Retry Scope stops and no further action attempts are performed, leaving `1` as the displayed output. This behavior follows the Retry Scope execution model: it runs the action, checks whether the condition has been met, and repeats only while the condition remains unsatisfied and retry limits have not been reached. The activity is designed for operations that may need repeated attempts, with the condition determining when the scope has succeeded. UiPath documents that the condition is

evaluated after the action, which is essential for determining both the number of executions and the output produced by activities inside the action. See the UiPath Retry Scope activity documentation and the UiPath control-flow documentation for the execution behavior of retry-based workflows.

QUESTION NO: 39

In UiPath Robotic Enterprise Framework, the value of MaxRetryNumber in the Config.xlsx file should be set to a number greater than 0 to enable the retry mechanism in the following cases:

Options are :

A. Get data from UiPath Orchestrator queues with Auto Retry disabled.

B. Get data from spreadsheets, databases, email, web API.

C. Do not work with UiPath Orchestrator queues.

ANSWER: B C

Explanation:

“Get data from spreadsheets, databases, email, web API.” and “Do not work with UiPath Orchestrator queues.” are correct because `MaxRetryNumber` is the retry setting used by Robotic Enterprise Framework when its transaction source is not an Orchestrator Queue. This includes processes that obtain transaction data from local spreadsheet rows, database records, email messages, or responses supplied by an external web service. In these implementations, the framework manages the retry count itself: after a transaction fails with an application exception, it can retrieve and process the same transaction again until the configured maximum has been reached. Consequently, setting the value higher than zero enables this framework-level retry behavior for non-queue transaction sources. The setting is also applicable as the general rule whenever the process does not use Orchestrator queues, regardless of the specific non-queue source selected. Queue-based processes instead rely on the queue item lifecycle and retry configuration maintained in Orchestrator. UiPath’s documentation describes the framework as supporting both queue and non-queue transaction sources and documents queue retry behavior separately. See UiPath Robotic Enterprise Framework documentation and UiPath queue retries documentation.

QUESTION NO: 40

What is the purpose of the WaitForReady property in any UiAutomation activity?

Options are :

A. Specifies to continue executing the remaining activities even if the current activity failed.

B. Specifies the amount of time (in milliseconds) to wait for the activity to run before an error is thrown.

C. Before performing the actions, waits for the target to become ready.

ANSWER: C

Explanation:

Before performing the actions, waits for the target to become ready. is correct because `WaitForReady` controls how a UI Automation activity checks that its target application or UI element is in a usable state before the activity carries out its operation. This synchronization step helps ensure that the robot does not try to click, type, select, or otherwise interact with an element while the application is still loading or processing.

UiPath provides readiness modes for this property, such as `None`, `Interactive`, and `Complete`. The selected mode determines the level of readiness verification performed before execution. For example, a more complete readiness check is useful when automation must wait for the application and its UI content to finish loading, whereas a lighter check can be appropriate when the target is known to be available quickly. Properly configuring this property improves reliability by synchronizing automation actions with the target UI rather than relying on fixed delays. UiPath documents this behavior in its

UI Automation activities documentation and describes UI-target synchronization concepts in the Studio activities documentation.

QUESTION NO: 41 - (SIMULATION)

In this exercise, you will create a UiPath automation that performs the steps below.

To achieve this, you will use the REFrameWork as the starting template and follow the UiPath development best practices.

The solution has to be scalable, so create two separate projects (sub-processes):

- One for the Dispatcher (add to queue);

- Another one for the Performer (consume queue).

Make sure you use a connection to an UiPath Orchestrator for testing.

Here are the steps performed by the Robot in the Dispatcher:

1.Login to https:/Nvww.acme-test.com.

2. On the landing page, Dashboard, click or hover over the Invoices menu item and then click on Search for Invoice. Click on Display All Invoices.

3. Scrape the data from the whole table displayed.

4. For each row in the datatable, Add a queue item containing the Invoice Number, Vendor TaxID and Date.

5. Close ACME System 1.

Here are the steps performed by the Robot in the Performer:

1.Login to https:/Nvww.acme-test.com.

2. For each Queue Item:

- Click or hover over the Invoices menu item and then click on Search for Invoice;

- Type the Invoice Number retrieved from the queue item into the Invoice Number field field;

- Click on Search;

- Extract the values for the Vendor TaxID and Date and compare them with the values from the queue item (check for EXACT match for all fields!);

- If the values are not matching, this should be categorized as a Business Rule Exception, and the queue em should have the status set accordingly;

- If the values match, the transaction is successful.

Note: Navigation can be achieved in multiple ways by the robot - choose whichever you find best.

Constraints to follow in the development, using the REFrameWork:

1. TransactionItem datatype should be a QueueItem. The process should recover and retry 2 times in case of errors in navigation between the Invoice Search and Invoices - Search Results pages. One transaction is the action of navigating to the Invoices Search page, searching for the Invoice Number and scraping the values from the resulting one row table.

2. Create a separate workflow file for the Login to ACME. File input arguments: URL; Username ; Password .

3. Create a separate workflow file for closing ACME.

4. Add the ACME_URL and ACME_Credential to the Excel Config file.

5. Populate InitAllApplications.xaml from the Framework folder with Invoking the Login to ACt...1E and navigation to the Work Items.

6. Populate CloseAllApplications.xaml from the Framework folder with Invoking the Close ACtv1E.

7.Populate KillAllProcesses xaml from the Framework folder with killing the process used.

8. Populate the Process.xaml file with the following actions:Navigation, Searching for Invoice Number, Scraping, Checking if the values match, Handling the Business Rule Exception.

Important Note: Don ' t use external file references outside of the project folder (including Orchestrator Assets). Place all the used files within the project folder, zip that folder and upload it to the UiPath Certification Platform.

Zip Al the used workflow files AND the output Excel fife. Then upload the.zip file to the UiPath Certification Platform.

Good luck!

ALL THE BEST!

ANSWER: See the explanation for the answer

Explanation:

Required deliverable: create and zip two independent UiPath REFramework projects: a **Dispatcher** project that produces queue items and a **Performer** project that consumes them. Both projects must be configured with a test UiPath Orchestrator connection and use the same Orchestrator queue.

1. Dispatcher project (REFramework)

Add the ACME URL and credential configuration entries to the project `Data/Config.xlsx` (for example, `ACME_URL`, `ACME_Credential`, and a queue-name setting such as `InvoiceQueueName`). Keep required project files inside the project folder.
Create `LoginACME.xaml` with input arguments `in_URL As String`, `in_Username As String`, and `in_Password As SecureString` (or the required password type used by the project). It opens ACME, navigates to the login page, enters the credentials, and logs in.
Create `CloseACME.xaml` to close the ACME browser/application.
In `Framework/InitAllApplications.xaml`, invoke `LoginACME.xaml` using the values read from Config and navigate to the invoice/work-items area as required.
Navigate through **Invoices → Search for Invoice → Display All Invoices**. Use `Extract Table Data` (or equivalent table scraping) to retrieve the *entire* displayed invoices table into a DataTable.
For every DataRow, use `Add Queue Item` for the configured queue. Set `SpecificContent` with exactly these values:

```
{
  "InvoiceNumber" : row("Invoice Number").ToString,
  "VendorTaxID"   : row("Vendor TaxID").ToString,
  "Date"          : row("Date").ToString
}
```

Invoke `CloseACME.xaml` from `Framework/CloseAllApplications.xaml`.
In `Framework/KillAllProcesses.xaml`, kill the browser/process used for ACME, such as `chrome`, `msedge`, or the selected browser process.

2. Performer project (REFramework)

Use the same Config values and queue name as the Dispatcher. Create the same reusable `LoginACME.xaml` and `CloseACME.xaml` workflows, or place equivalent copies inside this separate project.
In `InitAllApplications.xaml`, invoke the login workflow and perform the initial navigation to the required ACME work/invoice area. In `CloseAllApplications.xaml`, invoke `CloseACME.xaml`. In `KillAllProcesses.xaml`, kill the selected browser process.
Set the REFramework transaction-item type to `UiPath.Core.QueueItem`. In `GetTransactionData.xaml`, obtain the next item with `Get Transaction Item` from the invoice queue.
In `Process.xaml`, read the queue values using:

```
in_TransactionItem.SpecificContent("InvoiceNumber").ToString
in_TransactionItem.SpecificContent("VendorTaxID").ToString
in_TransactionItem.SpecificContent("Date").ToString
```

For each QueueItem, navigate to **Invoices → Search for Invoice**, enter the queued invoice number, click **Search**, and scrape the Vendor TaxID and Date from the one-row search-result table.

Compare Vendor TaxID and Date using exact string comparison (including the expected date representation). The invoice number is the search key; the returned Vendor TaxID and Date must exactly equal the queue-item values.
If either value differs, throw a `BusinessRuleException`. REFramework will call `Set Transaction Status` with **Business Exception**, so the QueueItem receives the Business Exception status and is not retried as a system failure.
If the values match, complete normally. REFramework will set the QueueItem status to **Successful**.
If navigation between Invoice Search and Invoice Search Results fails, let/throw a normal system exception rather than a BusinessRuleException. Configure the queue/REFramework retry behavior so that navigation system failures are retried **2 times** (maximum retry count 2). Thus, navigation failures are retriable application/system exceptions, while data mismatches are business exceptions.

Final packaging: verify that both projects run against the test Orchestrator queue, include each project's workflows, `Config.xlsx`, and the required output Excel file within the submitted content, and upload a ZIP that contains no references to files outside the project folder.

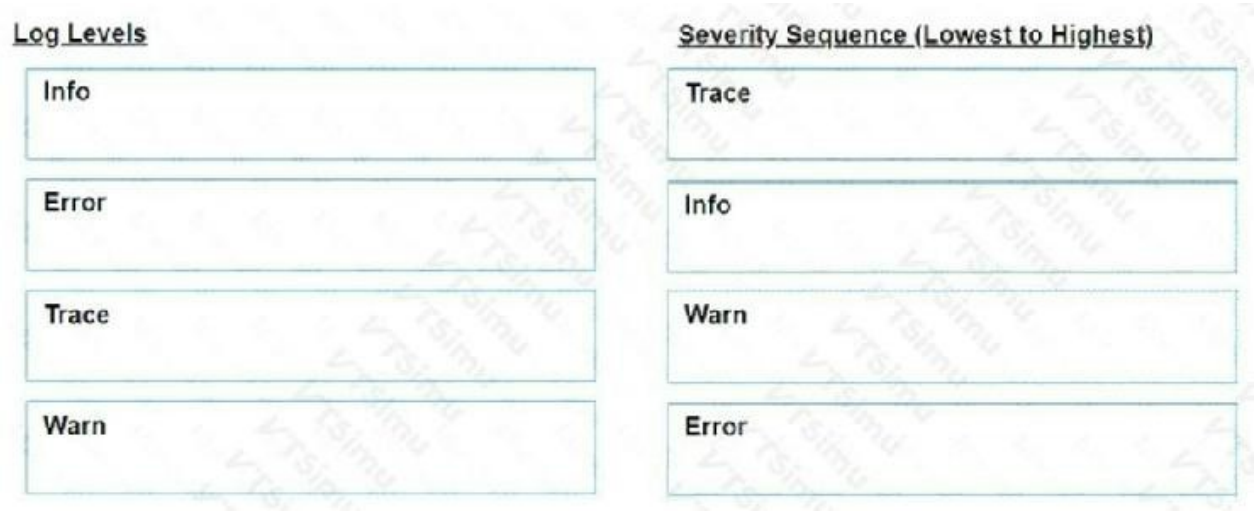
QUESTION NO: 42 - (DRAG DROP)

A developer is automating a project for the Finance team. As defined in the Process Definition Document, the robot is required to log the completion of each step with the documented corresponding Log Level.

If Fatal is the highest Log Level severity for the UiPath Log Message activity, what is the correct priority sequence, from the lowest to the highest severity, of the remaining Log Levels?

Instructions: Drag and drop the lowest severity level to the top of the list and repeat until the highest severity level appears at the bottom of the list.

ANSWER:



Explanation:

UiPath log levels are intended to make the importance of process events clear in execution logs, so that routine diagnostic information can be separated from conditions requiring attention or indicating a failure. The severity progression used by the Log Message activity begins with **Trace**, which is the least severe level and is primarily suited to detailed diagnostic or step-by-step execution information. Next is **Info**, used for normal process milestones, such as recording that a business step has completed successfully. **Warn** follows, representing an unexpected or notable condition that should be visible to reviewers but does not necessarily prevent the automation from continuing. **Error** is more severe and records a failure or condition that affects the expected processing of the automation. Above these is Fatal, which the question explicitly establishes as the highest severity level.

Because the requested list runs from lowest to highest and excludes Fatal, the slots must be filled top to bottom as **Trace**, **Info**, **Warn**, and **Error**. This ordering allows Finance-team log reviewers to interpret events consistently: detailed execution tracing comes first, ordinary completion records follow, then warnings requiring awareness, and finally errors reflecting failures. UiPath's logging documentation describes the available log levels and their severity behavior, including the progression through Trace, Info, Warn, Error, and Fatal. See the official UiPath documentation on logging for the Log Message activity's logging concepts and level definitions.