**Disclaimer**: All the points and questions answered here are based on my understanding of the topics, and even though it’s extremely low, there is a slight probability that some information may be wrong. Hence please crosscheck and do your own due diligence.

Actions

Three types of actions: Screen Action (Have Input parameter and local variable), Client Action (Have Input parameter and local variable and Output Parameters) and Server Action (Have Input parameter and local variable and Output parameter)

Screen Action can call other Screen Actions, Client Action and Server Action.

Client Action can call other Client Actions and Server Action.

Server Action can call other Server Actions.

But the reverse call can’t happen.

e.g., server action can’t call a screen action or client action, or client action can’t call a screen action.

Reactive programming Model:

Total 5 event calls.

4 are for screens (On Initialize>On Ready>On Render>On Destroy)

1 is for Aggregates/Data action (On After Fetch).

On render is called multiple times on screen whenever underlying data on screen or block changes.

On destroy event is called on the current screen to destroy the data of the previous screen. This happens only after the data on the current screen is fully rendered.

Blocks and Events

Blocks and screens have different scopes hence they don’t know of the changes occurring inside each other. To tell each other of the changes occurring inside their scope they pass messages to each other using input parameter that we called events. It happens to maintain the proper functionality.

A block triggers an event (Using trigger event) and parent handle it (Event handler).

Scope of trigger event is within block and event handler is within parent screen.

The parent must mandatorily handle the event using event handler if the trigger event of the block is mandatory.

Two types of events:

Trigger event: Tells parent of change inside the block scope

On Parameter Change event: Tells the Block of change inside the parent scope.

Entity Relationships:

One to One One-to-Many Many-to-One (Learn how they are created)

Primary key ------is called Identifier/Id in Outsystems

Foreign Key ------ is called Reference Attribute in Outsystems.

Not mandatory for entities to have an Id (Primary key) but to create relations between multiple entities we need it.

Id can be of type: Text, Long Integer and Other entity identifier.

Also learn about Structure and Static entities and their properties.

Like static entities can’t be changed during runtime and acts like enumeration, they have attributes and records.

Default Records: Id, Label, Order, Is Active. And each have identifier: Low, Medium, High.

Aggregates and Data Actions:

CRUD Operations (Actions): Create R(Get) Update Delete (In outsystems it’s “get” not “retrieve”)

Both data actions and aggregates are used to fetch data, they start executing when the screen is initializing and run asynchronously in parallel (Hence don’t have a particular order in which the run).

In aggregate u can add multiple filters and sorting methods.

In sorting if there are multiple attributes u must define the ascending and descending criteria for each.

The aggregate is first sorted based on the first attribute and its parameter, if there are multiple same rows in the first attribute then those are again sorted based on the second attributes and sorting parameter.

e.g., Table sorted on Name Ascending and Age Descending. (The people with same name will be later sorted on the age parameter in descending way)

Calculated aggregates:

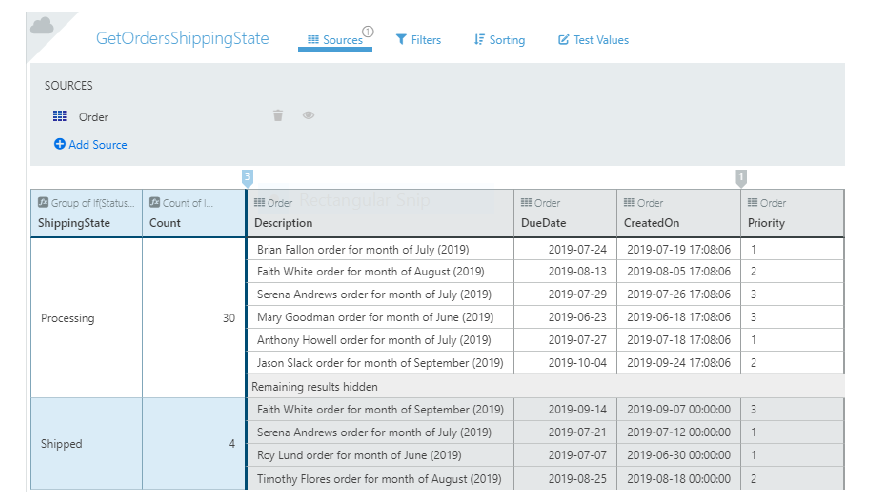
Let’s say there is an entity with two attributes “first name” and “last name” and u created a third attribute “full name” = “first name” + “last name” in that case the output of the aggregate would contain all three columns

FirstName Last Name Full Name (First Name + Last Name)

Aggregation Function:

Unlike in calculated attribute if you use aggregation function such as average, sum the output would only contain the new columns that you created and have single value as the output.

e.g., There is an entity with Player name and their runs. And you want to find total runs all the players have scored together and their average. This time you would use sum and average over the run attribute. which will add all the runs of the players and find their average, and that will be the only output of the aggregate the total runs and average runs.



In above image we have used group by and count hence the output would only contain first two columns.

Joins:

Only with (Inner join)

With or without (Left Join)

With (Outer join)

You are creating a one-to-many relation b/w two entities, if the reference attribute is mandatory an inner join will be created and if it’s not mandatory a left join will be created.

In advance SQL queries YOU USE BRACKETS AND OTHER SIGNS like:

{Entity} [Attribute] and @Inputparameter (for dynamic query)

SELECT{Person}. [Name] from {Person} where {Person}. [Department] = @Department

Delete Rules: Delete, Ignore, Protect.

For advance SQL query you must create the structure of the output that u expect the query to return. Even if it’s a non-select query.

Other Important points:

Site Property: Stored on server, not frequently changed, Modified in service centre.

Application property: Stored on browser, can be changed frequently.

Both have them are of basic datatypes like long integer or text which is same as that of identifier, hence we can say that they can be of type identifier.

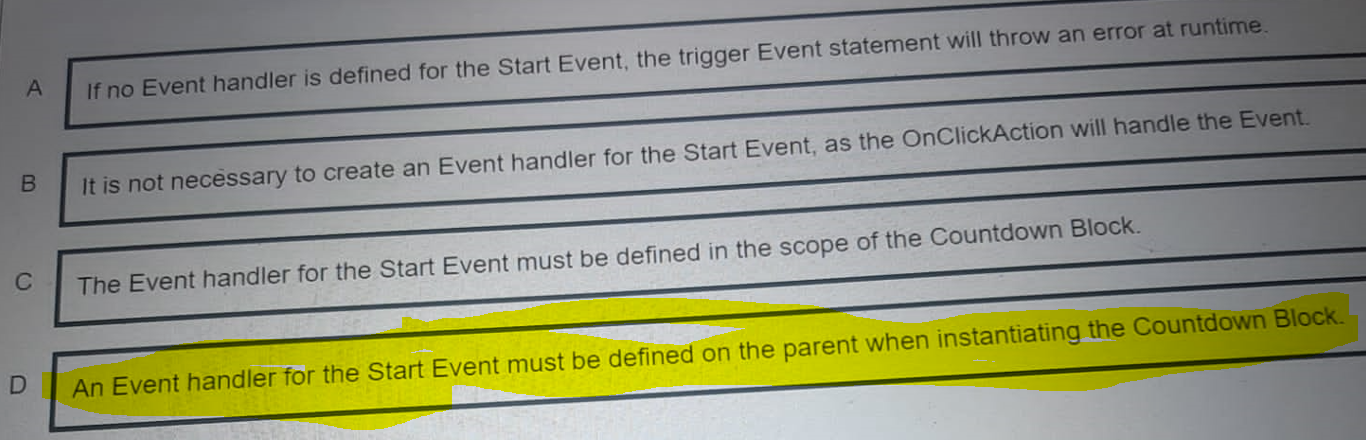
How to observe and answer the questions:



First always look for what the question is asking?? Here it’s asking for **correct option.**

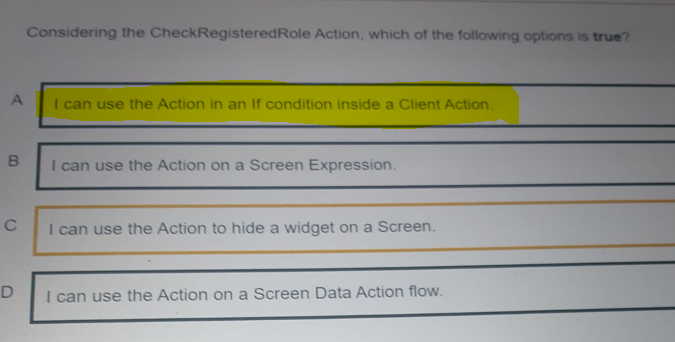
After that look at the other information provided in the fields like in the is mandatory it’s set to yes.

Now based on that we know that for the above trigger event there must be an event handler in the parent’s scope.



And option **D** is the most suitable one for this.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

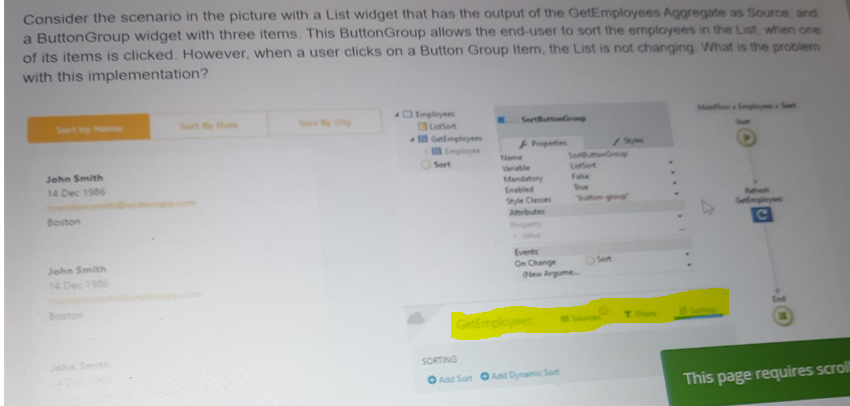


Now look at the above question, A check registered role action just returns a Boolean value as we know.

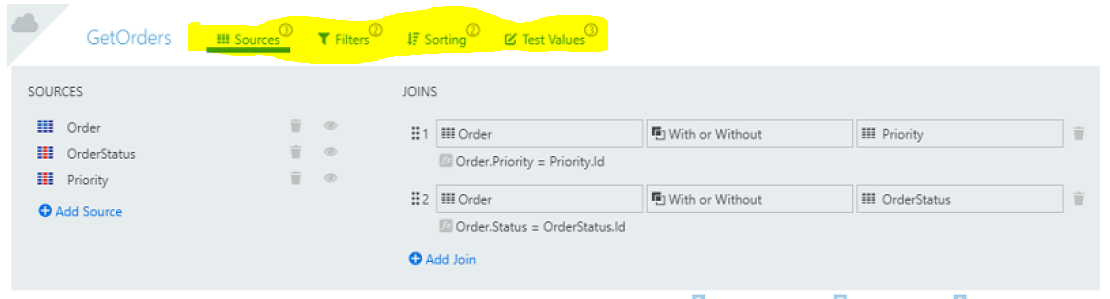
But that Boolean value we can use in an if else block to show a widget or hide a widget.

Option C appears correct but it’s not, as this action can’t hide or show a widget, it’s the if widget that hides or show the other widgets, hence option **A** is the correct one.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

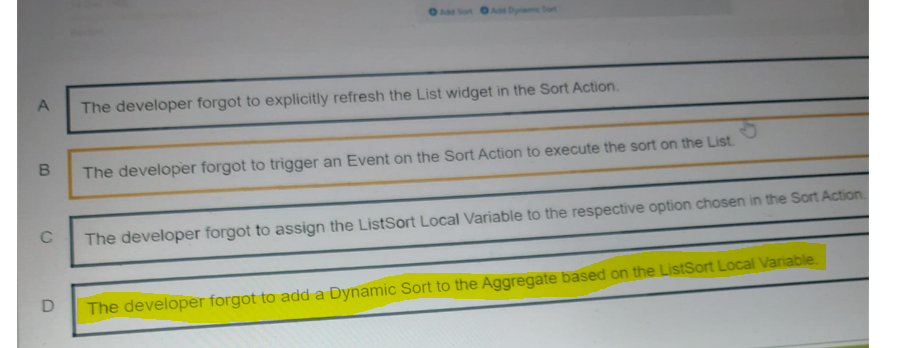


First look at the above image and read the question, now look at the yellow highlighted area of the aggregate, there are Sources, Filter and Sorting.

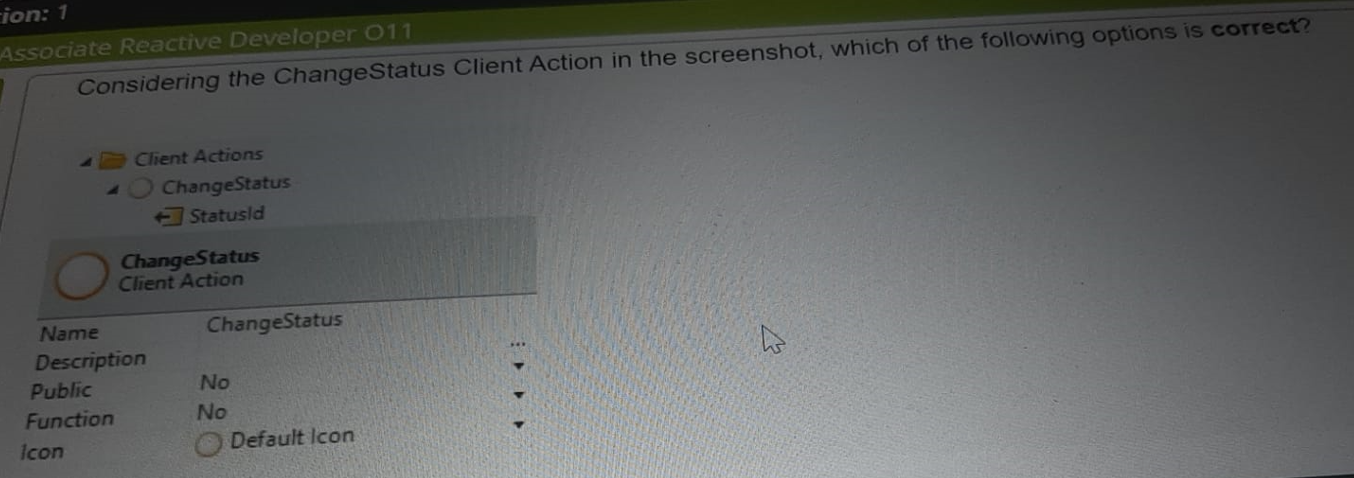


Now look at the proper implementation above and notice the bubble at the corners of filter and sorting with a number inside that shows how many filters and sorting criteria are defined.

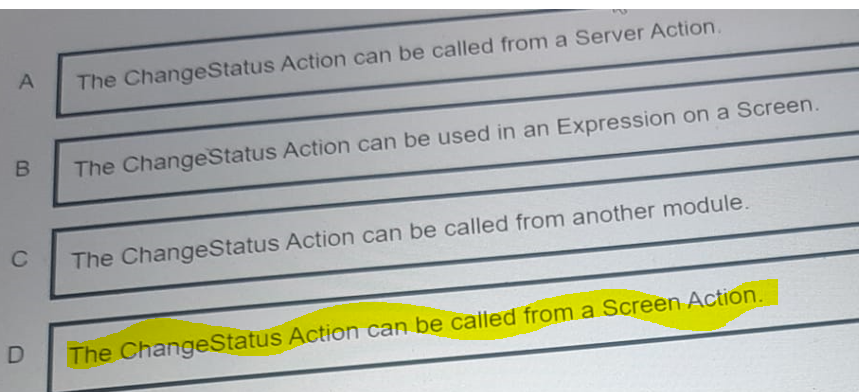
In the image with question there isn’t any bubble at the top corners which suggests there are no filter or sorting criteria defined by the user in that aggregate. Now based on that which option is the most suitable one below??

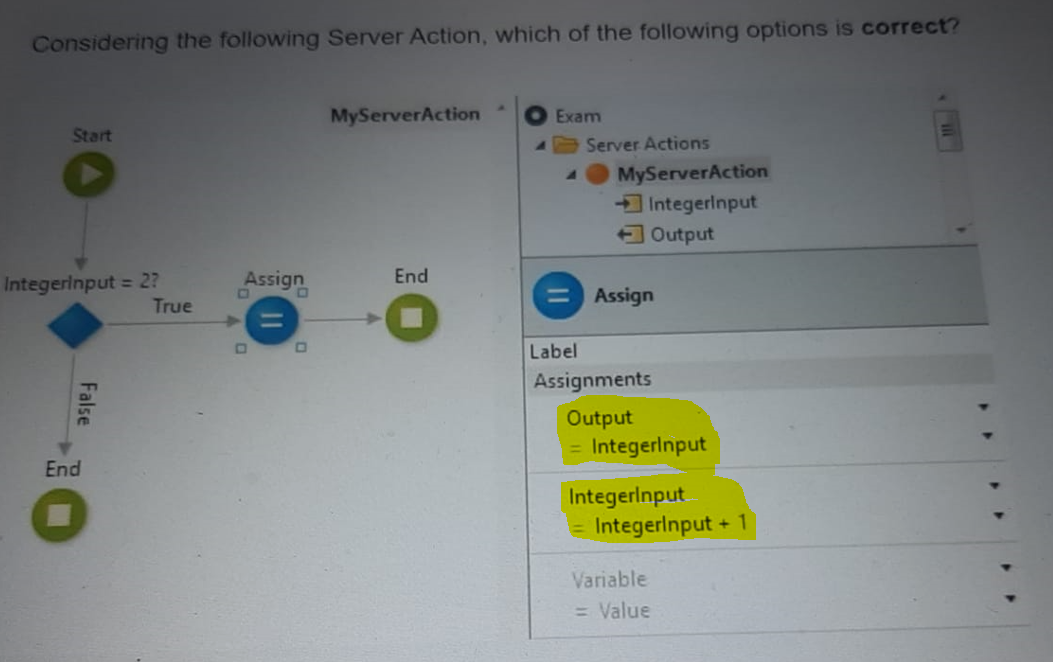


\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



In the above image we can see that public property is set to No, which means its scope is restricted to the module and it’s a **Client Action**. Also, a client action can be only called by another client action or a screen action Hence the correct answer is.

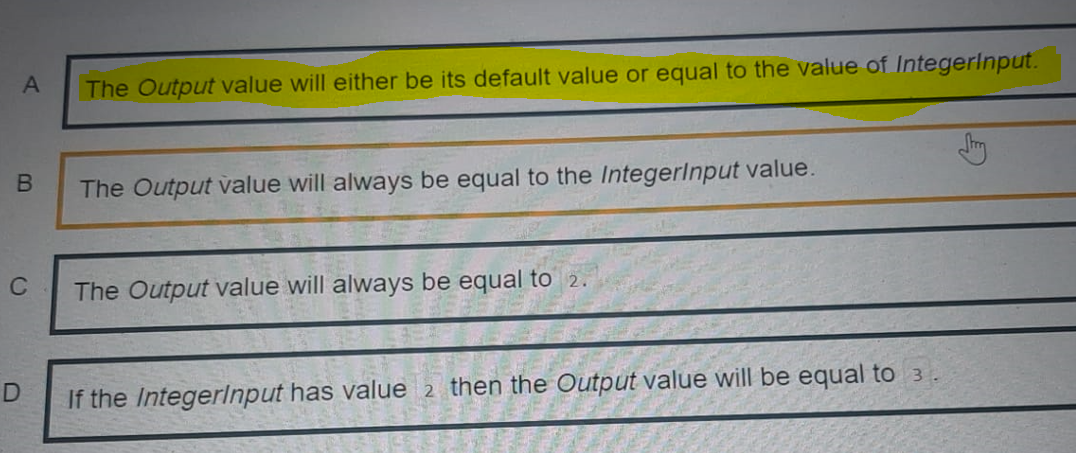


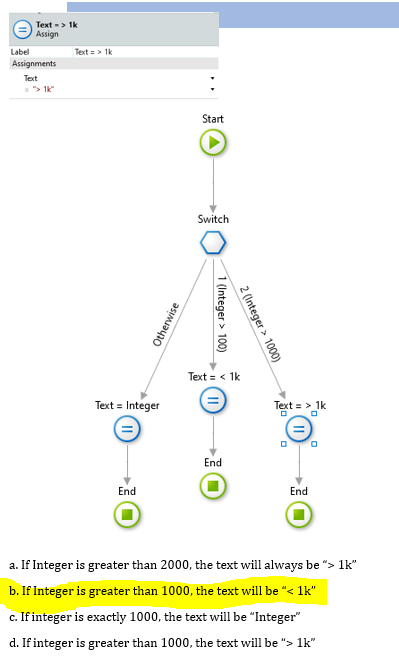


In above question Output is only assigned to Integer input in the true branch, which means if the condition is false the Output will be equal to its default value.

And looking at the order in which the Output is assigned to Integer Input it won’t be affected by the second assignment where the value of Integer Input is incremented by one.

Hence the output is either default value or 2(Integer Input)





Now look at the above question and see why the b option is correct.

In a switch the first condition that becomes true is executed, the first condition Integer>100

Hence if the value is 1500 it will be executed.

And if that’s executed look at the assign statement, it’s assigned <1000 even though it’s greater than 1000.

