Code segments

```
MSITokenProvider(". . .", null)

tp.GetAccessTokenAsync(". . .")

AzureServiceTokenProvider()

StringTokenProvider("storage", "msi")

tp.GetAuthenticationHeaderAsync(CancellationToken.None)
```

Answer Area

```
var tp = AzureServiceTokenProvider()

var t = new Token( tp.GetAccessTokenAsync(". . .")

return new StorageCredentials(t);
```

QUESTION 12
You need to ensure receipt processing occurs correctly. What should you do?

A. Use blob metadata to prevent concurrency problems.
B. Use blob SnapshotTime to prevent concurrency problems.
C. Use blob leases to prevent concurrency problems.
D. Use blob properties to prevent concurrency problems.

Correct Answer: B
Explanation:
You can create a snapshot of a blob. A snapshot is a read-only version of a blob that's taken at a point in time. Once a snapshot has been created, it can be read, copied, or deleted, but not modified. Snapshots provide a way to back up a blob as it appears at a moment in time.

Scenario: Processing is performed by an Azure Function that uses version 2 of the Azure Function runtime. Once processing is completed, results are stored in Azure Blob Storage and an Azure SQL database. Then, an email summary is sent to the user with a link to the processing report. The link to the report must remain valid if the email is forwarded to another user.

References:
https://docs.microsoft.com/en-us/rest/api/storageservices/creating-a-snapshot-of-a-blob

QUESTION 13
You need to construct the link to the summary report for the email that is sent to users. What should you do?

A. Create a SharedAccessBlobPolicy and add it to the containers SharedAccessPolicies. Call GetSharedAccessSignature on the blob and use the resulting link.
B. Create a SharedAccessBlobPolicy and set the expiry time to two weeks from today. Call GetSharedAccessSignature on the blob and use the resulting link.
C. Create a SharedAccessAccountPolicy and call GetsharedAccessSignature on storage account and use the resulting link.
D. Create a SharedAccessBlobPolicy and set the expiry time to two weeks from today. Call GetSharedAccessSignature on the container and use the resulting link.

Correct Answer: D
Explanation:
Scenario: Processing is performed by an Azure Function that uses version 2 of the Azure Function runtime. Once processing is completed, results are stored in Azure Blob Storage and an Azure SQL database. Then, an email summary is sent to the user with a link to the processing report. The link to the report must remain valid if the email is forwarded to another user.

Create a stored access policy to manage signatures on a container's resources, and then generate the shared access signature on the container, setting the constraints directly on the signature.

Code example: Add a method that generates the shared access signature for the container and returns the signature URI.

static string GetContainerSasUri(CloudBlobContainer container)

{

//Set the expiry time and permissions for the container.

//In this case no start time is specified, so the shared access signature becomes valid immediately.

SharedAccessBlobPolicy sasConstraints = new SharedAccessBlobPolicy();

sasConstraints.SharedAccessExpiryTime = DateTimeOffset.UtcNow.AddHours(24);

sasConstraints.Permissions = SharedAccessBlobPermissions.List | SharedAccessBlobPermissions.Write;

//Generate the shared access signature on the container, setting the constraints directly on the signature.

string sasContainerToken = container.GetSharedAccessSignature(sasConstraints);

//Return the URI string for the container, including the SAS token.

return container.Uri + sasContainerToken;

}

References:
https://docs.microsoft.com/en-us/azure/storage/blobs/storage-dotnet-shared-access-signature-part-2

QUESTION 14
HOTSPOT
You need to ensure that security requirements are met.

What value should be used for the ConnectionString field on line DB03 in the Database class? To answer, select the appropriate options in the answer area.
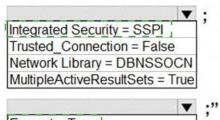
NOTE: Each correct selection is worth one point.

"Data Source=datastore.database.windows.net;Initial Catalog=expense;

▼ ;
Integrated Security = SSPI
Trusted_Connection = False
Network Library = DBNSSOCN
MultipleActiveResultSets = True

▼ ;"
Encrypt = True
Integrated Security = True
Failover Partner = False
Named Pipes = True

Correct Answer:

"Data Source=datastore.database.windows.net;Initial Catalog=expense;

▼ ;
Integrated Security = SSPI
Trusted_Connection = False
Network Library = DBNSSOCN
MultipleActiveResultSets = True

▼ ;"
Encrypt = True
Integrated Security = True
Failover Partner = False
Named Pipes = True

QUESTION 15
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this question, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You need to ensure that the SecurityPin security requirements are met.

Solution: Configure the web application to connect to the database using the WebAppIdentity security prinicipal. Using the Azure Portal, add Data Masking to the SecurityPin column and exclude the WebAppIdentity service principal.

Does the solution meet the goal?

A.   Yes
B.   No

Correct Answer: A

Topic 3, Proseware, Inc

Background
You are a developer for Proseware, Inc. You are developing an application that applies a set of governance policies for Proseware's internal services, external services, and applications. The application will also provide a shared library for common functionality.

Requirements
Policy service
You develop and deploy a stateful ASP.NET Core 2.1 web application named Policy service to an Azure App Service Web App. The application reacts to events from Azure Event Grid and performs policy actions based on those events.

The application must include the Event Grid Event ID field in all Application Insights telemetry.

Policy service must use Application Insights to automatically scale with the number of policy actions that it is performing.

Policies
Log policy
All Azure App Service Web Apps must write logs to Azure Blob storage. All log files should be saved to a container named logdrop. Logs must remain in the container for 15 days.

Authentication events
Authentication events are used to monitor users signing in and signing out. All authentication events must be processed by Policy service. Sign outs must be processed as quickly as possible.

Policylib
You have a shared library named PolicyLib that contains functionality common to all ASP.NET Core web services and applications. The Policy Lib library must

- Exclude non-user actions from Application Insights telemetry.
- Provide methods that allow a web service to scale itself.
- Ensure that scaling actions do not disrupt application usage.

Other
Anomaly detection service
You have an anomaly detection service that analyzes log information for anomalies. It is implemented as an Azure as a web service.

If an anomaly is detected, an Azure Function that emails administrators is called by using an HTTP WebHook.

Health monitoring
All web applications and services have health monitoring at the /health service endpoint.

Issues
Policy loss
When you deploy Policy service, policies may not be applied if they were in the process of being applied during the deployment.

Performance issue
When under heavy load, the anomaly detection service undergoes slowdowns and rejects connections.

Notification latency
Users report that anomaly detection emails can sometimes arrive several minutes after an anomaly is detected.

App code
EnventGridController.cs
Relevant portions of the app files are shown below. Line numbers are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

```
EventGridController.cs
EG01  public class EventGridController : Controller
EG02  {
EG03    public static AsyncLocal<string> EventId = new AsyncLocal<string>();
EG04    public IActionResult Process([FromBody]) string eventsJson
EG05    {
EG06      var events = JArray.Parse(eventsJson);
EG07
EG08      foreach (var @event in events)
EG09      {
EG10        EventId.Value = @event ["id"].ToString();
EG11        if (@event["topic"].ToString().Contains("providers/Microsoft.Storage"))
EG12        {
EG13          SendToAnomalyDetectionService(@event["data"]["url"].ToString());
EG14        }
EG15
EG16        {
EG17          EnsureLogging(@event["subject"].ToString());
EG18        }
EG19      }
EG20        return null;
EG21    }
EG22    private void EnsureLogging(string resource)
EG23    {
EG24      . . .
EG25    }
EG26    private async Task SendToAnomalyDetectionService(string uri)
EG27    {
EG28      var content = GetLogData(uri);
EG29      var scoreRequest = new
EG30      {
EG31        Inputs = new Dictionary<string, List<Dictionary<string, string>>>()
EG32        {
EG33          {
EG34            "input1",
```