Question: 1

Regarding memcache which of the options is an ideal use case?

A. Caching data that isn't accessed often
B. Caching data that is written more than it's read
C. Caching important data that isn't persisted to disk
D. Caching published data like blogs, news feeds, and other content that is read but not modified

Answer: D

Explanation:
Memcache is an App Engine feature that provides in-memory, temporary storage. It is provided primarily as a cache for rapid retrieval of data that is backed by some form of persistent storage, such as Google Cloud Datastore. Ideal use cases for Memcache include caching published data like blogs, news feeds, and other content that is read but not modified.
Refence:
https://cloud.google.com/appengine/articles/best-practices-for-app-engine-memcache#introduction

Question: 2

When you use Cloud Endpoints, what URL pattern is used in the config file to map to your code?

A. /_ah/spi/.*
B. Endpoints are automatically mapped
C. This is up to the developer
D. /_ah/api/.*

Answer: A

Explanation:
Endpoints listens on /_ah/spi/.* and serves up to users at /_ah/api
Refence: https://cloud.google.com/appengine/docs/python/endpoints/required_files

Question: 3

App Engine Standard Environments allow different runtimes. Two of them are Python 2.7 and Java 7. Which other runtimes are supported?

A. Go and Ruby
B. Go and PHP
C. PHP and Ruby

D. Swift, Rust and Ruby

Answer: B

Explanation:
The App Engine standard environment is based on container instances running on Google's infrastructure. Containers are preconfigured with one of several available runtimes (Java 7, Python 2.7, Go and PHP). Each runtime also includes libraries that support App Engine Standard APIs. For many applications, the standard environment runtimes and libraries might be all you need.
Refence: https://cloud.google.com/appengine/docs/about-the-standard-environment

Question: 4

A Datastore transaction can operate on a maximum of _____ entity groups?

A. 15
B. 20
C. 30
D. 25

Answer: D

Explanation:
An entity group is a set of entities connected through ancestry to a common root element. The organization of data into entity groups can limit what transactions can be performed:
. All the data accessed by a transaction must be contained in at most 25 entity groups.
. If you want to use queries within a transaction, your data must be organized into entity groups in such a way that you can specify ancestor filters that will match the right data.
. There is a write throughput limit of about one transaction per second within a single entity group. This limitation exists because Datastore performs masterless, synchronous replication of each entity group over a wide geographic area to provide high reliability and fault tolerance.
Refence:
https://cloud.google.com/datastore/docs/concepts/transactions#transactions_and_entity_groups

Question: 5

Regarding Datastore, which definition best describes an ancestor query?

A. A query over a single entity group using the key of a parent entity
B. A global query that filters based on a list of keys
C. A query that returns only keys
D. A query that returns multiple entity groups

Answer: A

Explanation:
Ancestor Query: A query over a single entity group using the key of a parent entity. By default, the results of such a query are strongly consistent.
Refence: https://cloud.google.com/appengine/docs/python/glossary#ancestor_query

Question: 6

If a free application hits a quota limit for a given resource what happens?

A. You'll automatically be updated to a paid application and be required to pay for overages
B. You cannot use that resource until the quota is replenished
C. You will get an email warning you that you're at your limit and need to address it
D. App Engine will disable your application

Answer: B

Explanation:
App Engine tracks your application's resource usage against system quotas. App Engine resets all resource measurements at the beginning of each calendar day (except for Stored Data, which always represents the amount of datastore storage in use). When free applications reach their quota for a resource, they cannot use that resource until the quota is replenished. Paid apps can exceed the free quota until their spending limit is exhausted.
Daily quotas are replenished daily at midnight Pacific time. Per-minute quotas are refreshed every 60 seconds.

Question: 7

Of the options given, which one is a valid reason for memcache to evict a value?

A. Attempting to read a record from a different service than it was created with
B. None. Memcache will only evict records after the expiration time
C. Memory pressure
D. App Engine's garbage collection

Answer: C

Explanation:
Memcache contains key/value pairs. The pairs in memory at any time change as items are written and retrieved from the cache.

By default, values stored in memcache are retained as long as possible. Values can be evicted from the cache when a new value is added to the cache and the cache is low on memory. When values are evicted due to memory pressure, the least recently used values are evicted first.
Refence:
https://cloud.google.com/appengine/docs/python/memcache/#Python_How_cached_data_expires

Question: 8

When using the configuration option for enforcing that a non-admin user must be logged in for a given handler, what value needs to be set?
login: _____

A. required
B. admin
C. mandatory
D. optional

Answer: A

Explanation:
When a URL handler with a login setting other than optional matches a URL, the handler first checks whether the user has signed in to the application using its authentication option. If not, by default, the user is redirected to the sign-in page. You can also use auth_fail_action to configure the app to simply reject requests for a handler from users who are not properly authenticated, instead of redirecting the user to the sign-in page.
Note: the admin login restriction is also satisfied for internal requests for which App Engine sets appropriate X-Appengine special headers. For example, cron scheduled tasks satisfy the admin restriction, because App Engine sets an HTTP header X-AppEngine-Cron: true on the respective requests. However, the requests would not satisfy the required login restriction, because cron scheduled tasks are not run as any user.
Refence: https://cloud.google.com/appengine/docs/python/config/appref

Question: 9

Of the options given which command will deploy an application to App Engine?

A. appcfg.py app deploy
B. appcfg.py deploy
C. appcfg.py push
D. appcfg.py update .

Answer: D

Explanation:

After you deploy, your application runs at the URL https://<YOUR-PROJECT-ID>.appspot.com.
To upload your application files, run the appcfg.py command with the update action and the name of your application's root directory. The root directory must contain the app.yaml file for the application.
appcfg.py update myapp/
Refence:
https://cloud.google.com/appengine/docs/python/tools/uploadinganapp#deploying_an_app

Question: 10

Regarding Identity and Access Management (IAM), which type of special account belonging to your application allows your code to access Google services programmatically?

A. OAuth
B. Code account
C. Simple Key
D. Service account

Answer: D

Explanation:
A service account is a special Google account that can be used by applications to access Google services programmatically. This account belongs to your application or a virtual machine (VM), instead of to an individual end user. Your application uses the service account to call the Google API of a service, so that the users aren't directly involved.
A service account can have zero or more pairs of service account keys, which are used to authenticate to Google. A service account key is a public/private keypair generated by Google. Google retains the public key, while the user is given the private key.
Refence: https://cloud.google.com/iam/docs/service-accounts

Question: 11

Regarding memcache, to ensure language-independence which data types should the different runtimes use for keys?

A. In Python use plain strings (not Unicode strings), In Java use byte arrays (not strings), In Go use byte arrays, In PHP use strings
B. In Python use Unicode strings, In Java use byte arrays (not strings), In Go use byte arrays, In PHP use strings
C. In Python use plain strings (not Unicode strings), In Java use int, In Go use int arrays, In PHP use strings
D. In Python use Unicode strings, In Java use strings, In Go use byte arrays, In PHP use strings

Answer: A

Explanation:
An App Engine app can be factored into one or more modules and versions. Sometimes it is convenient to write modules and versions in different programming languages. You can share the data in your memcache between any of your app's modules and versions. Because the memcache API serializes its parameters, and the API may be implemented differently in different
languages, you need to code memcache keys and values carefully if you intend to share them between langauges.
Key Compatibility
To ensure language-independence, memcache keys should be bytes:
In Python use plain strings (not Unicode strings)
In Java use byte arrays (not strings)
In Go use byte arrays
In PHP use strings
Refence:
https://cloud.google.com/appengine/articles/best-practices-for-app-engine-memcache#sharing_memcache_between_different_programming_languages

Question: 12

Regarding App Engine Services, which of the following statements is TRUE?

A. Services can each have a different runtime and performance settings
B. Services are no longer a part of App Engine
C. Services must use the same runtime
D. Services can each have a different runtime, but must have the same performance settings

Answer: A

Explanation:
At the highest level, an App Engine application is made up of one or more services, which can be configured to use different runtimes and to operate with different performance settings.
Refence: https://cloud.google.com/appengine/docs/python/an-overview-of-app-engine

Question: 13

Regarding the Users API, which primitive roles are considered an admin role?

A. Admin, Owner, Editor
B. Admin and Owner
C. Admin, Editor, Owner, Viewer
D. Editor, Owner, Viewer

Answer: D

Explanation:
While a user is signed in to an app, the app can access the account's email address for every request the user makes to the app. The app can also access a user ID that identifies the user uniquely, even if the user changes the email address for her account.
The app can also determine whether the current user is an administrator for the app. An admin user is any user that has the Viewer, Editor, or Owner primitive role, or the App Engine App Admin curated role. You can use this feature to build administrative features for the app, even if you don't authenticate other users. The Go, Java, PHP and Python APIs make it easy to configure URLs as "administrator only".
Refence: https://cloud.google.com/appengine/docs/python/users/

Question: 14

App Engine standard environments use an application configuration file to set application level settings. What is the name of that file?

A. app.yaml for Python, Go, and PHP, appengine-web.xml (Java)
B. app.config for all platforms
C. appcfg
D. app.config for Python, Go and PHP, app.config.xml for Java

Answer: A

Explanation:
You can configure your App Engine application's settings in the app.yaml (appengine-web.xml for Java) file. This file specifies how URL paths correspond to request handlers and static files. The app.yaml file also contains information about the application code, such as the application ID and the latest version identifier.
Refence: https://cloud.google.com/appengine/docs/python/config/appref

Question: 15

Of the options given which will allow App Engine services to share state?

A. None. Services are isolated
B. Shared instance memory
C. Datastore and Memcache
D. Datastore and Shared instance memory

Answer: C

Explanation:
Every service, version, and instance has its own unique URI, for example, v1.my-service.my-app.appspot.com. Incoming user requests are routed to an instance of a particular service/version according to URL addressing conventions and an optional customized dispatch file.

You can also pass requests between services and from services to external endpoints using the URL Fetch API.
All the services in an application share the state of the Datastore and Memcache services. They can also collaborate by assigning work between them to Task Queues. To access these shared services, use the corresponding App Engine APIs. Calls to these APIs are automatically mapped to the application's namespace.
Refence:
https://cloud.google.com/appengine/docs/python/an-overview-of-app-engine#communication_between_services

Question: 16

App Engine autoscaled instances have a request timeout of how long?

A. Unlimited
B. 3600 seconds
C. 60 seconds
D. 120 seconds

Answer: C

Explanation:
App Engine autoscaled instances have a 60-second deadline for HTTP requests, 10-minute deadline for tasks.
Refence:
https://cloud.google.com/appengine/docs/python/an-overview-of-app-engine#scaling_types_and_instance_classes

Question: 17

Regarding Cloud IAM, you can use the console to grant a role to a team member. Which other options are also valid?

A. The gcloud tool or the applyPolicy() method
B. The group policy admin page or the federated security dashboard
C. The gcloud tool or the setIamPolicy() method
D. The App Engine SDK or the setIamPolicy() method

Answer: C

Explanation:
Project owners can grant access to team members to access project's resources and APIs by granting IAM roles to team members. You can grant a role to a team member using the Cloud Platform Console, the gcloud tool, or the setIamPolicy() method.

Refence:
https://cloud.google.com/iam/docs/granting-changing-revoking-access#granting_access_to_team_members

Question: 18

Regarding Datastore, if you want to pass a key to the back-end in a GET or POST request, what is the recommended way?

A. Manually hashing a key
B. Calling the url_safe method of a key in Python Calling KeyFactory.keyToString in Java
C. Calling the makeSafe method of a Key for Python and Java
D. Manually salting and hashing a key

Answer: B

Explanation:
You can also use an entity's key to obtain an encoded string suitable for embedding in a URL:
url_string = sandy_key.urlsafe()
This produces a result like agVoZWxsb3IPCxIHQWNjb3VudBiZiwIM which can later be used to reconstruct the key and retrieve the original entity:
sandy_key = ndb.Key(urlsafe=url_string)