THEORY
First Part (Units 1-4)

1. In a distributed system:

A	There is a single agent or process. False. A distributed system should consist of at least two agents or processes.
B	Agents are independent and they do not cooperate. False. Agents should cooperate, otherwise they are strictly independent processes that aren't members of any wider system.
C	Multiple agents cooperate and try to achieve a common goal, since distributed systems are examples of concurrent systems. True. A distributed system should consist of multiple cooperating agents. Every distributed system is also a concurrent system.
D	All agents share their state. False. Each agent has its own state and is able to take several decisions based on that local state.
E	All the above.
F	None of the above.

2. Some characteristics of cloud computing are:

A	Provision of elastic services. True. The services being provided in the cloud should be scalable and adaptive. Those two characteristics define elasticity.
B	Without failure transparency. False. Cloud computing is implemented in distributed systems. Every distributed system should provide failure transparency.
C	Single-threaded implementations are mandatory. False. Cloud computing components may be implemented using either multi-threaded agents or event-based agents.
D	Use of a P2P interaction approach. False. Most cloud computing services use a client-server interaction approach. P2P interaction might be also used, but this is not something that characterises cloud computing. Its usage is only marginal in cloud computing.
E	All the above.
F	None of the above.

3. In the SaaS cloud service model:

A	The main resources being provided to the customers are virtual machines. False. The SaaS service model provides complete applications to its customers, instead of "virtual machines".
B	Customers should deal with application deployment. False. Customers in the SaaS model only USE applications. They don't need to deploy them. They have been deployed by the cloud provider.
C	Customers should deal with application development. False. Customers in the SaaS model only USE applications. They don't need to develop them. They have been developed by the cloud provider.
D	Services aren't scalable. False. Those applications being provided should be scalable.
E	All the above.
F	None of the above.

4. In the PaaS cloud service model:

A	The main resources being provided to the customers are virtual machines. False. The PaaS service model provides a programming platform with automatized service management to its customers, instead of "virtual machines".
B	Customers should deal with application deployment. False. Customers in the PaaS model develop applications. They don't need to deploy them. They should be deployed by the cloud provider.
C	Customers should deal with application development. True. Customers in the PaaS model develop applications and request their deployment.
D	Services aren't scalable. False. Those services being provided should be scalable.
E	All the above.
F	None of the above.

5. Examples of cloud computing service models:

A	X-Window.
B	NFS.
C	TCP/IP.
D	SOA.
E	All the above.
F	None of the above. The cloud computing service models are: SaaS, PaaS and IaaS.

6. High availability is a required property in every distributed system. This means that:

A	Service developers do not need to worry about availability, since failures are always masked by the operating system. False. Operating systems are unable to hide all possible process- or communication-related failures.
B	Service developers and providers should use appropriate mechanisms (e.g., replication) in order to achieve failure transparency. True. The statement given above implies that the service USERS shouldn't perceive failures. Because of this, failures should be hidden by the service developers with the help of the middleware or underlying protocols. The latter might cooperate in that failure handling.
C	Neither faults nor failures happen in the components of distributed services. We shouldn't worry about them. False. Unfortunately, component faults and component failures may always happen in a distributed system. Note that all "anomalous conditions" (i.e., faults) may not be forecast.
D	The assumed system model avoids that failures occur in our developed services. False. Every system model should choose a given failure model: the one closest to its actual behaviour. All failure models consider that failures may arise.
E	All the above.
F	None of the above.

7. System models are needed because:

A	With them, we may analyse the correctness of algorithms and protocols. True. This is the main target of system models.
B	They ensure data consistency. False. System models provide a high-level view of system characteristics, centred in the most relevant aspects of component behaviour. Those aspects condition the design of algorithms and protocols. They cannot ensure data consistency. We need consistency-specific protocols to this end.
C	They improve service throughput. False. System models provide a high-level view of system characteristics, centred in the most relevant aspects of component behaviour. Those aspects condition how algorithms and protocols should be designed. They don't deal with throughput.
D	They mask faults, errors and failures. False. System models provide a high-level view of system characteristics. They don't manage nor mask failures.
E	All the above.
F	None of the above.

8. Multi-threaded servers...

A	Need concurrency control mechanisms in order to avoid race conditions. True. When multiple concurrent activities try to access any shared resource demanding exclusive access, concurrency control mechanisms are needed. Multi-threaded servers are examples of this kind.
B	Are highly scalable since their activities never block. False. Since concurrency control mechanisms are needed, such mechanisms may block requesting threads when the shared resource has been already assigned to another thread. So, those activities may block and this reduces the scalability of multi-threaded servers.
C	Considering a single process, they implement sequential execution of tasks in that process, instead of concurrent execution. False. Their execution is concurrent.
D	Assuming a single multi-threaded server, its multiple threads always ensure high availability in case of process crash failures. False. When the process crashes, all its threads also crash. So, high availability cannot be ensured by a multi-threaded server.
E	All the above.
F	None of the above.

9. Advantages of asynchronous servers (AS) when they are compared with multi-threaded servers (MTS):

A	AS increase the degree of concurrency. False. Each AS process has one thread in most cases, while MTS have multiple threads per process. So, the degree of concurrency is higher in MTS.
B	AS do not manage nor need external events. False. AS are event-based. So, they should manage both internal and external events.
C	AS implement decentralised algorithms, MTS cannot implement them. False. Decentralised algorithms may be implemented using both kinds of servers.
D	AS transparently use replication mechanisms, MTS cannot use them. False. Replication mechanisms are used in the same way by both kinds of servers.
E	All the above.
F	None of the above.

10. Communication events in a distributed system model are...

A	...internal events. False. Internal events are those generated in a process without relating it with any other component. As such, communication events cannot be internal, they are external events. External events are those that relate an agent with its environment. They may be input events (e.g., message reception) or output events (e.g., message sending).
B	...agents. False. Agents are the entities that execute actions associated to events. Communication events are a subset of external events. Agents may originate communication events, but communication events aren't agents.
C	...failure models. False. A failure model specifies which kind of failures may happen in a system. They aren't communication events.
D	...external events. True. See the explanation of the first option.
E	All the above.
F	None of the above.

11. REST is an example of:

A	System model.
B	Failure model.
C	Replication protocol.
D	Replication model.
E	All the above.
F	None of the above. False. As it is discussed in the student guide of Unit 3, REST is an "architectural style" for web-based distributed services. It is explained as an example of remote method invocation mechanism / subsystem. So, it isn't a system model nor failure model nor replication protocol nor replication model.

12. SOAP is an example of:

A	SaaS. False. SaaS is the acronym of "Software as a Service" and it is an example of cloud service model. SOAP isn't an example of service.
B	Replication protocol. False. Replication protocols take care of the steps being needed in order to replicate a given service. SOAP is the acronym of Simple Object Access Protocol. It is an example of invocation protocol used at the middleware layer. It doesn't require replicated components. Besides this, invocation is only a step in service management. Replication protocols consider many other aspects (update propagation, concurrency control, failure detection, recovery of replicas, addition of replicas…). So, SOAP isn't a valid example of replication protocol. It is only an example of middleware protocol related to service invocation.
C	IaaS. False. IaaS is the acronym for "Infrastructure as a Service" and it is an example of cloud service model. SOAP isn't an example of service.
D	Middleware protocol. True. Take a look at the explanation given in the second option.
E	All the above.
F	None of the above.

13. Middleware goals:

A	Interoperability. True. Middleware components use standard protocols in order to promote their interoperability (and that of the applications developed on top of them) with other software components.
B	To provide high-level solutions to common problems. True. Again, its reliance on standards generates thoroughly tested mechanisms that provide valid solutions to problems that may arise in many distributed applications.
C	Transparency. True. Many aspects of distribution transparency can be solved at the middleware layer. For instance, location transparency is achieved using RPC (remote procedure calls) or ROI (remote object invocations). Both mechanisms may be used in middleware products.
D	Reliability. True. Since middleware use standard protocols and those protocols have been specified by expert software developers, middleware-layer solutions are (or, at least, should be) reliable.
E	All the above.
F	None of the above.

14. ZeroMQ is a messaging middleware with...

A	Transient communication. False. ZeroMQ provides weakly persistent (i.e., non-transient) and asynchronous communication. It is weakly persistent since ZeroMQ doesn't use message brokers. Brokers are generally used for providing communication persistency. Instead of this, it is implemented as a library that uses buffers in each one of the participating processes. With such help, it can only provide weakly persistent communication.
B	Weakly persistent communication. True. See above.
C	Synchronous communication. False. It provides asynchronous communication.
D	Remote method invocation support. False. It is a middleware centred on message-oriented communication. This implies that such middleware is not object-oriented. Remote method invocation only makes sense in systems that manage objects.
E	All the above.
F	None of the above.

15. Broker-based messaging middleware systems implement:

A	Synchronous communication. False. The usage of a broker in a messaging middleware is only related to its degree of persistency not to its degree of synchrony.
B	Heartbeat-based failure detection. False. Failure detection is not a problem being solved by brokers in messaging middleware products.
C	Asynchronous communication. False. The usage of a broker in a messaging middleware is only related to its degree of persistency not to its degree of synchrony.
D	Persistent communication. True. The goal of general broker-based communication systems it to provide communication persistency.
E	All the above.
F	None of the above.

16. Some deployment-related tasks are:

A	Software development. False. Deployment tasks deal with software installation, configuration and reconfiguration stages. Development is a stage that should be completed before deployment.
B	Software design. False. Deployment tasks deal with software installation, configuration and reconfiguration stages. Design is a stage that should be completed before deployment.
C	Software analysis. False. Deployment tasks deal with software installation, configuration and reconfiguration stages. Analysis is a stage that should be completed before deployment.
D	Software upgrades. True. Deployment tasks deal with software installation, configuration and reconfiguration stages. Software upgrades is a stage that belongs to the reconfiguration chapter. So, it is included in the deployment-related set of tasks.
E	All the above.
F	None of the above.

17. Containers:

A	Host software components.
B	Isolate and protect components from external accesses.
C	Manage some component life cycle events.
D	Implement dependency injection.
E	All the above. All previous options summarise in a precise way which are the main goals of software containers.
F	None of the above.

18. IaaS directly provides the following deployment tasks:

A	Component development. False. As it has been already explained in previous questions, component development (in general, software development) is not a deployment-related task.
B	VM allocation. True. This is the main goal in an IaaS service model: to manage VM provision.
C	Component upgrades. False. Component upgrades aren't directly supported by an IaaS provider. They are the responsibility of IaaS customers. They are supported and automatized in the PaaS service model.
D	Scaling out decisions. False. Horizontal scalability isn't directly supported by IaaS providers. These decisions are the responsibility of IaaS customers. They are supported and automatized in the PaaS service model.
E	All the above.
F	None of the above.

19. A fault domain is:

A	A middleware protocol.
B	A failure model.
C	A system model.
D	A replication model.
E	All the above.
F	None of the above. A fault domain is a concept that has been presented when the Azure support was described in Unit 4. It refers to a set of HW resources that depends on the same fault sources. A PaaS customer needs to consider fault domains in order to request the deployment of some service component replicas in different fault domains. Thus, the deployed service may overcome failures, since they usually happen in a single fault domain at a time. Despite being a concept related to faults (and, indirectly, to failures) it is not a failure model. Such description also shows that it isn't a middleware protocol, a system model nor a replication model.

20. A Service Level Agreement (SLA)...

A	Specifies the Quality of Service (QoS) to be provided by a service once deployed. True. This is an informal definition of SLA.
B	Should be considered by a PaaS provider in order to automatize its scaling decisions. True. The SLA is used by PaaS providers in order to know when they should take their scaling decisions. To this end they compare the current performance metrics with the goals stated in the SLA. When such current performance is poor, additional service instances need to be added (scaling out). On the contrary, when the current performance values are far better than the goals, some existing instances may be dropped in order to reduce costs (scaling in).
C	Usually specifies the availability to be assured by a given deployed service. True. One of the regular service quality metrics is availability. It is considered in most SLAs.
D	Is considered by the monitoring and reaction subsystems in a PaaS in order to manage elastic services. True. This is a consequence of what has been described in option B.
E	All the above.
F	None of the above.

Second Part (Units 5-9)

21. About faults, errors and failures:

A	A failure without component redundancy generates an error. False. The correct sentence is: "An error without component redundancy generates a failure."
B	A fault is the manifestation of an error in a system. False. The correct sentence is: "An error is the manifestation of a fault in a system".
C	An error is an anomalous physical condition. False. That is the informal definition of a fault, instead of an error.
D	When errors are detected and appropriately handled, generate failures. False. Detection and appropriate management prevent faults from becoming errors. If those same mechanisms are applied onto errors, they won't be the cause of failures. Instead of this, those mechanisms might mask such existing errors.
E	All the above.
F	None of the above.

22. These sentences define each one of the following failure models:

A	In the stop failure model, processes fail showing an arbitrary behaviour. False. That is a short definition for Byzantine failures.
B	In the crash failure model, processes fail either crashing or receiving a subset of the messages sent to them. False. That is a compact definition for receive-omission failures.
C	In the general-omission failure model, processes fail either crashing or sending a subset of the messages they should send. False. That is a short definition for send-omission failures.
D	In the Byzantine failure model, processes fail halting and remaining in that state. False. That is a definition for stop failures.
E	All the above.
F	None of the above.

23. Replication...:

A	May improve process availability. True. If there are multiple copies of a given process or service, when one of them fails the other still remain available.
B	May introduce consistency problems. True. If there are multiple copies of a given data element, those copies cannot be updated at the same precise time. So, those copies aren't strictly consistent and some consistency issues may arise. Some update propagation algorithm is needed in order to forward those updates. Depending on the algorithm being used, we may comply with a different consistency model.
C	May enhance service performance. True. If most of the operations being served do only imply read actions, those operations may be served by a single replica. In that case, if a given service uses N replicas, it may improve its performance N times.
D	May introduce severe performance problems if most served operations generate state updates. True. This is the opposite case to what has been described in option C. When the operations being served update the service state, those updates should be propagated and applied to the other replicas. This may demand a lot of time if the network being used has a low bandwidth and a high latency. So, this might introduce serious performance problems in those cases.
E	All the above.
F	None of the above.

24. State-machine replication: NOTE: This question hasn't been considered in the exam assessment. Instead of "State-machine replication", it should have used "Active replication".

A	Ensures strict consistency. False. Strict consistency is almost impossible to ensure in a distributed system. State-machine replication, when it is appropriately implemented, may ensure that every replica receives the same sequence of requests. This may lead to a consistency similar to the sequential one, but not to strict consistency.
B	May overcome arbitrary failures. True. Arbitrary failures that affect a few replicas may be overcome comparing in the client proxy all answers being received from every replica and considering the answer being provided by a majority of them as the correct reply.
C	May lose request messages when the primary replica crashes. False. State-machine (or active) replication doesn't use "primary" replicas.
D	Needs synchronous communication to avoid failures. False. Failure avoidance doesn't depend on the degree of communication synchrony. It depends on the amount of failed replicas and on the failure scenario (i.e., whether the failure situation matches what was assumed in the failure model being considered to implement the replicated service).
E	All the above.9
F	None of the above.

25. Content coupling:

A	Is the best coupling level. False. It is the worst coupling level in the set presented in TSR.
B	Manages inter-module communication using simple parameters. False. That is a characteristic of "data coupling". In content coupling each component is able to access data contained in the internals of other components.
C	Minimizes inter-module communication. False. In this kind of coupling components freely access other components, without any encapsulation. This behaviour introduces a lot of communication, losing data access locality. It is the worst coupling level.
D	Minimizes component dependences. False. In this type of coupling there may be many component dependences.
E	All the above.
F	None of the above.

26. In order to improve the scalability of services based on persistent data, the following principles should be followed:

A	Increase concurrency as much as possible. False. Concurrency can be increased with care and only in cases where such concurrency doesn't introduce accesses to shared resources. That is not the general case: when we introduce concurrent activities in a given service, sooner or later they need to access the same resources. Those resources should be protected using concurrency control mechanisms. In that latter case, performance improvements are completely lost due to the long blocking intervals being introduced.
B	Use disks with low bandwidth. False. This introduces a severe bottleneck in disk accesses and it cannot improve service scalability.
C	Use a relational schema and the JOIN operator. False. Relational databases (i.e., those using a relational schema and its associated operators like the JOIN one) use pessimistic concurrency control mechanisms that lead to long blocking intervals. These databases are the best ones for ensuring programme correctness but, unfortunately, they aren't highly scalable.
D	Don't use long ACID transactions. True. The general recommendation is to avoid long transactions, since they might generate long blocking intervals.
E	All the above.
F	None of the above.

27. If we want a scalable service with strong consistency and high availability...

A	We should deploy that service in a data centre that avoids any network partition. True. Considering the CAP theorem, we are requesting strong consistency (C) and high availability (A). So, we should sacrifice network partition tolerance (P). To this end, we could try what is being described in this option.
B	We should assume a partitionable failure model when we design its algorithms and protocols. False. Network partition tolerance should be sacrificed as described above. To this end, we cannot use algorithms and protocols that assume that partitions may arise since, with them, strong consistency cannot be achieved.
C	We should deploy that service in multiple data centres in order to ensure its scalability. False. This introduces the risk of network partitions. If a partition arises in that scenario, either strong consistency is lost (if we still accept client requests in each data centre) or high availability cannot be ensured (if we prevent some of the data centres from answering client requests in order to ensure strong consistency in a single data centre, stopping all the others).
D	We should use at least two different replication models in order to ensure its high availability. False. High availability doesn't depend on the amount of replication models being considered, but on the other aspects of the CAP theorem: C and P.
E	All the above.
F	None of the above.

28. Bigtable is a...

A	Relational database management system. False. Bigtable is not an example of a relational database management system, but an example of NoSQL datastore.
B	IaaS system. False. Bigtable is not a cloud system that follows the infrastructure-as-a-service service model.
C	NoSQL datastore. True.
D	PaaS system. False. Bigtable is not a cloud system that follows the platform-as-a-service service model.
E	All the above.
F	None of the above.

29. In order to scale up a given deployed service, we may...

A	Change its replication model. False. Scaling up (i.e., vertical scalability) actions need to improve the hardware of the node being considered. To change a replication model only affects the algorithms being used; i.e., it is a software-related action.
B	Assuming that it was deployed in an IaaS, upgrade its VM type. True. If we move our service from a given VM type to a more powerful VM type we have improved the (virtual) hardware of our nodes. This is an example of vertical scalability.
C	Change its replication protocol. False. This is the same case explained in option A.
D	Assuming that it was deployed onto an IaaS, rent additional VMs and deploy other instances of its components on those VMs. False. These actions lead to improve horizontal scalability instead of vertical scalability.
E	All the above.
F	None of the above.

30. Throughput is improved using these mechanisms:

A	Network partitions. False. A network partition is a kind of failure. Failures don't improve throughput.
B	Asynchronous servers. True. Asynchronous programming tries to avoid all possible blocking sources in the execution of processes. To avoid blocking is one of the best mechanisms for improving throughput.
C	The arbitrary failure model. False. Failure models describe the most important aspects to be considered in failure management. This cannot improve throughput in a direct way. Indeed, when the arbitrary failure model is being assumed, programmers should deal with all possible answers from external components. This implies that the resulting programmes should have explicit sections of code for appropriately manage those situations. This doesn't improve throughput. In the general case, throughput will be worse (when we assumme an arbitrary failure model) than in other failure models.
D	Persistent storage. False. Persistency introduces additional delays when processes use the stored data. So, throughput cannot be improved using persistent storage.
E	All the above.
F	None of the above.

31. In cloud computing, elasticity is important because...

A	It reduces customer costs. True. Elasticity (scalability + autonomous and dynamic adaptability) is the key for implemeting a pay-as-you-go model in cloud computing. Due to this, customers only pay for those resources / services that they have used. This reduces their costs.
B	It minimizes the amount of provided resources. True. Elasticity is also convenient for cloud providers, since they may use scale down or scale in actions in order to reduce their costs when the workload being supported is low.
C	It doesn't compromise performance. True. Performance is maintained in good levels using elastic services.
D	It doesn't compromise the SLA. True. Elastic services may comply with the SLA quality requirements minimizing both provider and customer costs.
E	All the above.
F	None of the above.

32. Data distribution improves scalability because...

A	Workload is balanced. True. Instead of concentrating all workload in a single data repository being managed by a single node, data distribution allows spreading those data elements among multiple nodes, balancing the overall load among them.
B	Concurrency is increased without introducing contention. True. With a careful design, most data operations only require one of the resulting data shards. So, those accesses may be served by a single process. Since there are multiple shards, each server may concurrently serve a different request. This doesn't introduce additional contention (i.e., potential blocking) at the time of accessing such logically shared resource: the database.
C	May be combined with replication. True. Each one of the data shards may be replicated in order to improve its availability. One example of this was presented when we explained MongoDB: it combined sharding (i.e., database partitioning, a particular case of data distribution) with replication.
D	May be combined with asynchronous servers. True. Each one of those servers may be single-threaded and event-oriented. Nothing prevents this from happening. VoltDB is an example of modern database management system that combines sharding with asynchronous servers.
E	All the above.
F	None of the above.

33. A trusted computing base (TCB) should be...

A	Provided by external agents. False. It should be based on elements that are internal to the system being considered. Additionally, it shoud be minimal and simple.
B	Minimal and simple. True. Note that if any of its elements is compromised, the entire security system is broken. So, it should consist of as few elements as possible.
C	Composed by as many services as possible. False. This contradicts what has been explained in option B.
D	Based on the SLA. False. Current SLAs do not consist of security metrics. Note also that security is a qualitative attribute and this implies that there is no clear security-related value able to measure the level of quality in a security system.
E	All the above.
F	None of the above.

34. There are different types of security risks. Those types are:

A	Physical, authentication-related and authorisation-related. False. These are the types of security mechanisms.
B	Coarse, medium and fine. False. These are the types of defensive strategies.
C	Vulnerabilities, threats and attacks. True. These have provided the focus of discussion in Unit 9.
D	Unstructured, structured, internal and external. False. These are the types of security threats.
E	All the above.
F	None of the above.

35. There are different types of security mechanisms. Those types are:

A	Physical, authentication-related and authorisation-related. True. These are the types of security mechanisms.
B	Coarse, medium and fine. False. These are the types of defensive strategies.
C	Vulnerabilities, threats and attacks. False. These are the types of security risks.
D	Unstructured, structured, internal and external. False. These are the types of security threats.
E	All the above.
F	None of the above.

36. Some examples of malware are:

A	Denial of service. False. This is an example of security attack class.
B	Viruses and Trojans. True. Malware is one of the classes of security attacks. The others are: information gathering, access, and denial of service. Viruses and Trojans are examples of malware. Malware includes all software that provokes damage on the system where it is deployed. Other examples of malware are: worms and backdoors.
C	Man-in-the-middle. False. This is an example of security attack in the "access" class.
D	Port scans. False. This is an example of security attack in the "information gathering" class.
E	All the above.
F	None of the above.

37. Basic architectural patterns with unidirectional communication are...

A	PUSH-PULL.
B	Implicitly asynchronous.
C	PUB-SUB.
D	Easily scalable.
E	All the above. Unidirectional communication patterns are implicitly asynchronous since they don't require any answer from the other side (option B). As a result of this, they are easily scalable (option D). In ZeroMQ both the PUSH-PULL (option A) and PUB-SUB (option C) communication patterns (and their derived architectural patterns) are unidirectional.
F	None of the above.

38. When we use an intermediate queue in an advanced multi-client/multi-server architectural pattern...

A	The broker uses a REP socket as its front-end and a REQ socket as its back-end. False. The broker frontend socket should be of type ROUTER in order to remember the identity of the client. This is needed to appropriately forward the forthcoming reply. Such frontend may be complemented with a DEALER socket it its implicit round-robin distribution strategy is enough for passing the requests to the corresponding workers. On the other hand, when we want to select the intended worker (for instance, when we know the current load in each worker) we should use a ROUTER socket, again, as the backend. This second approach is the preferred one.
B	The broker uses a SUB socket as its front-end and a PUB socket as its back-end. False. See the explanation in option A.
C	The broker uses two ROUTER sockets: one as its front-end and another as its back-end. True. See the explanation in option A.
D	The servers use PUSH sockets. False. See the explanation in option A.
E	All the above.
F	None of the above.

39. Idempotent operations...

A	...are a synonym for "deterministic operations". False. Idempotent operations may be executed multiple times in a given server, generating always the same effect on its state. Deterministic operations are those that, with the same set of arguments and internal state values, provide in all invocations the same result. They are not identical concepts. Some operations may be deterministic (e.g., to withdraw a given amount from some account) since once we know its arguments and the state of the object where they will be applied we may easily forecast its result, but they aren't idempotent (if we call twice such an operation, the generated balance will be different in each call).
B	...are the only way to implement "at most once" semantics; i.e., we cannot implement that semantics with other types of operations. False. "At most once" semantics means that, in case of doubt about a request success, we shouldn't retry it. That behaviour doesn't depend on the kind of operation being invoked.
C	...are safe for implementing "at least once" semantics. True. "At least once" means that, in case of doubt about a request success, we should retry it. With idempotent operations there will be no danger when we repeat its execution (in case of success in the dubious attempt).
D	...are mandatory for implementing eventual consistency. False. Eventual consistency is easily implemented using commutative operations. Idempotent operations aren't commutative in the regular case.
E	All the above.
F	None of the above.

40. In the passive replication model described in the advanced client/server architectural patterns...

A	Multiple primary active replicas (for the same service) may work at once without problems. False. Only a primary replica should be running in a passive replication model.
B	A backup replica needs a client request in order to replace a crashed primary replica. True. The client request may break the tie in case of an incorrect suspicion about a primary failure. This happens when the backup replica doesn't receive any heart-beats from the primary one. If the primary has really crashed, clients will be unable to communicate with it. As a result of this, they will use the backup replica. When the backup receives those requests it knows for sure that the primary failed.
C	Clients aren't needed for replacing a crashed primary replica. False. See the explanation given in option B.
D	The primary replica never fails in that replication model. False. The primary replica may fail. Nothing prevents it from failing.
E	All the above.
F	None of the above.