2nd Partial, Theory (TSR)

This exam consists of 40 multiple choice questions. In every case only one of the answers is correct. Answers must be provided in a separate template. All questions have the same value. If correctly answered, they contribute 0.25 points to the final grade. If incorrectly answered, the contribution is negative, equivalent to 1/5th of the correct value; i.e., -0.05 points. So, think carefully your answers.

1. The Byzantine failure model...

A	...shouldn't be assumed for developing a real distributed service, since an actual set of computers cannot have that behaviour. False. It can be assumed without any problem. Processes may show an arbitrary behaviour in case of errors that lead to failures.
[✓ B]	...assumes that processes may have any arbitrary behaviour. True. That is the definition of Byzantine failures.
C	...doesn't make sense nowadays. As its name suggests, it represents the behaviour of ancient computers, instead of that of modern ones. False. This failure model makes sense. Processes may show an arbitrary behaviour in case of errors that lead to failures.
D	...assumes that processes may only fail by halting and process failures may be detected by other processes. False. That is part of the definition of stop failures.
E	All the above.
F	None of the above.

2. When we design an algorithm assuming a *stop* failure model...

A	...the resulting algorithm is usually simple since we assume that processes behave according to their specifications until they fail. True. In the *stop* failure model processes behave correctly until they stop. Once stopped they remain forever in that state. Such situation may be detected by other processes. With these assumptions, an algorithm may be designed without worrying about failures, since a failed process will not introduce any problems in the remaining processes. As a result of this, algorithms find their most favourable context when this failure model is assumed.

B	...we may find problems to implement such algorithm, since operating systems and middleware cannot guarantee a perfect behaviour for processes. True. When a process has errors and those errors lead to failures, the behaviour of that incorrect process is difficult to forecast and control. In the general case, it may send incorrect information to other processes that could not be easily managed by them. So, it is almost impossible to implement in a perfect way what is being assumed in this failure model.
C	...we assume that processes only fail by halting and such failures can be detected by correct processes. True. That is the definition of this failure model.
D	...we assume that communication channels behave correctly. True. In the classification proposed by Schneider, stop failures are the most benign failure model. Communication channels missbehaviour starts to be considered in the "crash+link" failure model, which is the third one in that classification. This means that communication problems do not arise in the stop model.
E	All the above.
F	None of the above.

3. There is a network partition failure when...

A	...a process is halted. False. Network partition failures are related with connectivity between sets of nodes. A process halt is not relevant in that context.
B	...a process has an arbitrary (i.e., outside its specifications) behaviour. False. Network partition failures are related with connectivity between sets of nodes. A Byzantine failure is not relevant in that context.
C	...a database is fragmented and distributed among multiple computers, but in an incorrect way (e.g., losing some rows in several tables). False. Network partition failures are related with connectivity between sets of nodes. How a database has been fragmented is not relevant in that context.
D	...the CAP theorem is not respected. False. The CAP theorem considers other aspects besides network connectivity. Additionally, what is stated in the CAP theorem cannot be broken. So, there is no way for by-passing what is stated in the CAP theorem.
E	All the above.
F	None of the above.

4. In the "primary partition" model:

A	Minor subgroups (i.e., those with less than a half of the system nodes) should be stopped. True. This partition managing model consists in only allowing execution in a major subgroup (considering as such a set of actively and correctly interconnected system nodes that encompasses more than a half of the total system nodes). As a result of this, every minor subgroup should be stopped when a network partition occurs.

B	All existing network channels behave as expected. False. If all network channels were correct, no network partition would have occurred. So, we couldn't talk about "primary partitions", since a primary partition model only makes sense in case of network partitions.
C	We are applying the CAP theorem, sacrificing partition tolerance. False. We are applying the CAP theorem, but sacrificing availability since minor subgroups are compelled to stop. Note that in this case we are providing strong consistency into the major subgroup and we are able to deal with a network partition situation (i.e., partition tolerance) but at the cost of preventing minor subgroups from continuing their work.
D	All node subgroups may go on. False. Minor subgroups should be stopped.
E	All the above.
F	None of the above.

5. Safety is...

A	...a qualitative attribute of dependability. False. It can be measured, since it is a probability: the probability that a system doesn't fail in a catastrophic way at time t if its was working correctly at time t'=0. Measurable aspects are quantitative attributes. The "qualitative" adjective refers to unmeasurable attributes.
B	...the probability S(t) that a distributed system is restored at time t if it had failed at time t'=0. False. That definition corresponds to maintainability.
C	...a failure model. False. It is a quantitative attribute of dependability.
D	...one of the aspects being considered in the CAP theorem. False. The CAP theorem considers consistency, availability and network partition tolerance, but not "safety".
E	All the above.
[✓ F]	None of the above.

6. Replication improves service performance when...

[✓ A]	...we use a ROWA (read one, write all) protocol and most operations only imply read accesses. True. We have assumed ROWA protocols in this subject. In a ROWA protocol each read-only operation should be served by a single replica, while write-related operations should finally affect all existing replicas, updating their state. In this kind of protocols, when most of the invoked operations are read-only, those requests may be distributed among all existing replicas, balancing their load and increasing the resulting performance.

B	...all the operations imply write accesses. False. Write operations demand that all existing replicas were involved in the operation service. In most cases this reduces overall performance, instead of improving it.
C	...the replicas are continuously recovering due to process failures. False. When there are failures, the system needs additional time to update the state of a recovering replica, finding out which have been its missed updates and propagating and applying those updates to such replica.
D	...a passive replication model is used, network connectivity is lost and state updates cannot be propagated to backup replicas. False. In that case the system should remain blocked in order to maintain its consistency. Otherwise, it should block only the disconnected backup replicas, allowing progress in the other one(s). But even in that second case, it will need additional efforts to recover their state when the connectivity loss is fixed. So, in the end, performance isn't increased in those situations.
E	All the above.
F	None of the above.

7. In the passive replication model:

A	Multiple replicas may receive and directly process any client request. False. This occurs in the active replication model, not in the passive one.
B	All replicas play the same role. False. This occurs in the active replication model, not in the passive one.
[✓ C]	In order to implement it, we cannot assume the arbitrary failure model. True. Arbitrary failures cannot be managed in the passive replication model. Arbitrary failure management is based on comparing the results provided by multiple active replicas, assuming that only a few of them may fail simultaneously. To this end, we need more than one primary-like replica, and this is not allowed in the passive model.
D	In order to implement it, we cannot assume the stop failure model. False. The stop failure model can be assumed without any constraint.
E	All the above.
F	None of the above.

8. About data-centric consistency models:

A	Causal is stricter than cache. False. The causal and cache consistency models cannot be compared. There are executions that are causal but not cache, and there are also executions that are cache but not causal.

B	Processor is stricter than causal. False. As in the previous case, these two consistency models cannot be compared.
C	Cache is stricter than FIFO. False. As in the first part, these two consistency models cannot be compared.
[✓ D]	Sequential is stricter than cache. True. Sequential is stricter than processor, and processor is stricter than cache. So, sequential is stricter than cache.
E	All the above.
F	None of the above.

9. Coupling measures...

[✓ A]	...the degree of dependency between the modules of an application. True. We have the best coupling when such coupling is loose (or weak). This means that the modules are only related by operations that have a minimal number of simple parameters. This reduces the need of interaction among modules.
B	...the reliability of an application. False. That is not the aim of coupling.
C	...service continuity. False. That is not the aim of coupling.
D	...whether each one of the dimensions considered in the CAP theorem is guaranteed. False. That is not the aim of coupling.
E	All the above.
F	None of the above.

10. Weak cohesion isn't convenient because...

A	...always leads to faults, errors and failures. False. Cohesion measures the correct association of functionality to each module. A module should be developed for providing a unique and clear functionality. Fault, error and failure metrics are not the objective of cohesion.
B	...implies availability loss. False. Cohesion measures the correct association of functionality to each module. A module should be developed for providing a unique and clear functionality. Failure evaluation or service availability are not the objective of cohesion.
[✓ C]	...it is unclear which is the functionality of each operation. This prevents modules from being reused in other applications. True. Cohesion measures the correct association of functionality to each module. A module should be developed for providing a unique and clear functionality. In case of weak cohesion, a given module or operation provides multiple functionalities, complicating its interfaces and reducing its reusability.
D	...ensures strong consistency among service replicas. False. Cohesion isn't related with inter-replica consistency.
E	All the above.
F	None of the above.

11. In a distributed service with weak coupling:

A	Request messages are generally small. True. We have the best coupling when such coupling is loose (or weak). This means that the modules are only related by operations that have a minimal number of simple parameters. This reduces the need of interaction among modules. The message being used for calling a remote operation are small in that case.
B	The amount of inter-component interactions is minimised. True. We have the best coupling when such coupling is loose (or weak). This means that the modules are only related by operations that have a minimal number of simple parameters. This reduces the need of interaction among modules.
C	There is a high degree of locality in data accesses. True. We have the best coupling when such coupling is loose (or weak). This means that the modules are only related by operations that have a minimal number of simple parameters. This reduces the need of interaction among modules. Most of the accesses are local in that case.
D	Each operation only requires a few arguments. True. We have the best coupling when such coupling is loose (or weak). This means that the modules are only related by operations that have a minimal number of simple parameters, requiring the same amount of arguments when they are called.
[✓ E]	All the above.
F	None of the above.

12. NoSQL datastores...

A	...ensure data consistency using ACID transactions. False. ACID transactions are only used in relational database management systems. They are not available in NoSQL datastores.
[✓ B]	...are usually more scalable than relational database management systems. True. Their variations onto relational database management systems were introduced for improving their scalability.
C	...do not ensure data persistence. False. They are an example of data persistence service.
D	...have a complex query language based on the “join” operator. False. The JOIN operator is a relational operator that has been dropped in NoSQL datastores.
E	All the above.
F	None of the above.

13. Key-value datastores...

A	...are examples of relational database management systems. False. They are an example of NoSQL datastores.

B	...use a schema based on objects with a variable number of attributes. False. That kind of schema is used in document datastores, not in key-value datastores. Key-value datastores only manage tables with two columns (or attributes). So, their number of attributes is static.
C	...examples are MongoDB and SimpleDB. False. They are examples of document datastores.
D	...examples are Cassandra and PNUTs. False. They are examples of extensible record datastores.
E	All the above.
[✓ F]	None of the above.

14. About extensible record datastores...

A	They use schemas based on tables with a variable number of columns. True. They use that kind of schema.
B	They use sharding for improving their scalability. True. Sharding (or database partitioning) is a very good mechanism for improving scalability. It is widely used in extensible record datastores.
C	One example is Bigtable. True.
D	One example is Cassandra. True.
[✓ E]	All the above.
F	None of the above.

15. About the CAP theorem...

[✓ A]	It relates consistency, availability and partition-tolerance. True. The CAP theorem relates these three properties, stating that only two of them may be ensured simultaneously in a distributed system. The other should be sacrificed.
B	Its result only makes sense in synchronous distributed systems. False. Such result doesn't depend on the degree of synchrony being assumed or managed in the system.
C	It states that strong consistency and high availability cannot be assured simultaneously. False. Those two properties can be assured if we renounce to network partition tolerance.
D	It relates data consistency, atomicity and persistency. False. It relates consistency, availability and partition-tolerance.
E	All the above.
F	None of the above.

16. About the CAP theorem consequences...

A	To develop a highly available and partition-tolerant service, we need to use eventual consistency.

	True. We should renounce to strong consistency. The usage of eventual consistency has been a common approach to deal with the CAP theorem implications when availability and network partition tolerance are the two selected properties.
B	To develop a strongly consistent (e.g., sequential) and highly available service we shouldn't tolerate network partitions. True. In that case we should deploy our service with care in several close nodes with redundant communication channels.
C	Eventual consistency is almost mandatory for scalable services, since they should be highly available and should overcome network partitions. True. Scalable services may need a large amount of nodes when their workload is high. To this end, they will be deployed in multiple datacentres and network partitions may arise. A service cannot be considered highly scalable if it isn't highly available. As a result, we should sacrifice strong consistency. Because of this, eventual consistency is the common choice in these cases.
D	The "primary partition" model is assumed in order to sacrifice availability, manage network partitions and assure strong consistency. True. This has already been explained in question 4C.
[✓ E]	All the above.
F	None of the above.

17. The three dimensions of scalability are...

A	...consistency, availability and partition-tolerance.
B	...reliability, security and safety.
C	...hardware, firmware and software.
D	...user interface, business logic and persistent data.
E	All the above.
[✓ F]	None of the above. The three dimensions of scalability are: scalability on size, scalability on distance and administrative scalability. So, none of the previous choices is correct.

18. Vertical scalability consists in...

A	...adapting the computing capacity to the current workload. False. Scalability with adaptability is elasticity.
B	...ensuring service continuity while the computer is upgraded. False. This refers to availability in the upgrading steps.
C	...increasing the amount of nodes where a service is running. False. That is horizontal scalability.
[✓ D]	...improving the computing capacity of a single node. True. This has been the traditional definition of vertical scalability.
E	All the above.
F	None of the above.

19. Horizontal scalability consists in...

A	...adapting the computing capacity to the current workload. False. Scalability with adaptability is elasticity.
B	...ensuring service continuity while the computer is upgraded. False. This refers to availability in the upgrading steps.
C	...increasing the amount of nodes where a service is running. True. That is horizontal scalability.
D	...improving the computing capacity of a single node. False. This has been the traditional definition of vertical scalability.
E	All the above.
F	None of the above.

20. Four complementary mechanisms for achieving scalability on size are...

A	...reliability, availability, maintainability and safety. False. These are four of the aspects of dependability. They are not directly related with scalability on size.
B	...task distribution, data distribution, replication and caching. True. These have been the four mechanisms presented in Unit 7 to this end. Tasks distribution and data distribution are able to balance the load among the processes that implement a given service. Replication is mandatory for ensuring the availability of the resulting service and for linearly improving its performance in case of read-only operations. Caching is a variant of replication started by clients. It reduces the need of interaction between clients and servers for recently obtained results.
C	...strict consistency, sharding, active replication and partition tolerance. False. Although sharding and general replication may improve scalability on size, strict consistency is too restrictive for implementing scalable services. Besides this, partition tolerance is not a mechanism for boosting scalability, but an aspect to be considered when we design algorithms and protocols in our systems.
D	...elasticity, cloud computing, grid computing and P2P computing. False. Cloud and grid are two distributed computing paradigms. P2P is an interacting approach. Elasticity is a goal. As such, none of those four concepts is a mechanism for implementing any kind of service (scalable or not).
E	All the above.
F	None of the above.

21. Properties of decentralised algorithms:

A	Failure of one process ruins the algorithm. False. One of the properties states just the opposite: "Failure of one process doesn't ruin the algorithm". Decentralised algorithms should be able to overcome failures. Even when failures arise, they should be able to continue immediately. Their reconfiguration in that case should be trivial.

[✓ B]	Processes take decisions based on local information. True. In order to take a decision each process should rely on its local information. Otherwise it might block for a long interval trying to collect remote information. Processes shouldn't be blocked in order to build scalable services.
C	Processes assume that a global clock exists. False. Processes in a decentralised algorithm should not depend on the current time. If any time-related decision should be taken, it should be taken considering only the local clock.
D	One process has complete information about the system state. False. When this happens, the failure of such process will be critical, becoming a single point of failure and provoking such algorithm failure.
E	All the above.
F	None of the above.

22. ***Sharding* enhances scalability because…**

A	It is a mechanism based on data distribution. True. Sharding consists in partitioning the data contained in a given database, distributing each of the resulting parts to a different node. So, it is a data distribution mechanism.
B	It is a mechanism that provides load balancing. True. As a result of its data distribution, the original workload is also distributed among the nodes that have received each one of the database parts, instead of being concentrated in a single server. With this, the load is balanced among multiple processes.
C	It increases the concurrency degree; i.e., the resulting service is able to process a large amount of concurrent requests. True. Each node that received a different database part is able to process a different request at the same time. With this the concurrency degree is increased as many times as parts exist.
D	When it is appropriately designed, it doesn't demand any synchronisation step. True. When the parts are appropriately chosen, each client query may be served by a single server process using the data contained in a single database part. In those cases, no message needs to be exchanged between server processes.
[✓ E]	All the above.
F	None of the above.

23. A service is elastic when…

A	...it is reliable and highly available. False. These are two of the properties demanded to a dependable service, but elasticity is not a synonym of dependability.

B	...it is dependable and uses a fast consistency model. False. Elasticity isn’t this.
C	...it is scalable and with dynamic and autonomous adaptability. True. Elasticity is the combination of scalability and adaptability.
D	...it is fault-tolerant and secure. False. As in the first element of this question, these two characteristics belong to dependable services but they are not what defines elasticity.
E	All the above.
F	None of the above.

24. In order to implement an elastic service, we need...

A	A monitoring system for the current workload.
B	A reactive system that automatizes service reconfiguration, taking scale-out and scale-in decisions.
C	A reactive system that considers the service level agreement.
D	A monitoring system for the current throughput.
E	All the above. The implementation of an elastic service by a PaaS provider demands two complementary subsystems: a monitoring one and a reactive one. The former should monitor the current workload and the current performance providing the needed input for the latter. The reactive subsystem analyses the differences between those two metrics and, based on them, takes the appropriate horizontal scalability actions. To this end, it should also consider the existing service level agreement. This means that all previous sentences are true.
F	None of the above.

25. The main goals of security are...

A	...to ensure data consistency and persistence. False. This is not related to security.
B	...secrecy, integrity, availability and accountability. True. These are the four security goals presented in Unit 8.
C	...safety, reliability, availability and maintainability. False. These are all the remaining aspects of dependability.
D	...transparency, service continuity, performance and efficiency. False. These are general goals of distributed services, but they aren’t the main goals of a security-related subsystem.
E	All the above.
F	None of the above.

26. The goal of a security policy is:

A	To ensure the correctness of a security system. False. This is the goal of security assurance.
B	To implement a security system. False. This is the goal of security mechanisms.
[✓ C]	To specify a security system. True. This is the goal of security policies.
D	To measure the dependability of a system. False. Global dependability metrics aren't the goal of security policies.
E	All the above.
F	None of the above.

27. About security mechanisms:

A	They are techniques and tools needed for implementing security. True. This is the definition of security mechanisms.
B	There are three main classes: physical, authentication-related and authorisation-related (i.e., related with access control). True. These are the classes of security mechanisms presented in Unit 8.
C	An example is the use of passwords. True. It is an example of authentication-related mechanisms.
D	An example is the file permission bits in UNIX file-systems. True. It is an example of authorisation-related mechanisms.
[✓ E]	All the above.
F	None of the above.

28. A security threat is:

A	A weakness in devices, protocols, programmes or policies in a system. False. That is a security vulnerability, not a security threat.

B	The probability T(t) that a system performs its functions at time t if it has been functioning correctly since time t'=0. False. This is the definition of reliability.
C	A model that specifies which divergences are allowed in the replicas of a given data element. False. This is the definition of replica consistency.
D	A set of rules that specifies which actions are authorised to the principals in a given system. False. This is possible definition for security policy.
E	All the above.
[✓ F]	None of the above.

29. Examples of security policy vulnerabilities:

A	Denial of service. False. It is an example of security attack class.
B	Man in the middle. False. It is an example of security attack in the access class.
C	SYN floods. False. It is an example of security attach in the denial of service class.
[✓ D]	Lack of disaster recovery plans. True. This is an example of security policy vulnerability.
E	All the above.
F	None of the above.

30. Examples of configuration vulnerabilities:

A	Unstructured threats. False. This is a type of security threat. Vulnerabilities aren't threats.
[✓ B]	To allow fragile passwords. True. This is an example of configuration-related vulnerability.
C	Packet sniffers. False. This is a kind of security attack in the information gathering class.
D	Phishing. False. This is a kind of security attack in the access class.
E	All the above.
F	None of the above.

31. About mechanisms in cryptographic protocols:

A	A MAC ensures non-repudiation. False. A message authentication code ensures integrity, but it cannot ensure non-repudiation. To this end, we need certificates.
B	A certificate is a mechanism for providing accountability. False. It ensures non-repudiation. Accountability is a goal in authorisation-related mechanisms.
C	One-way functions are used in both MAC and certificates. True. Messages authentication codes use one-way crypto-hashing functions and MACs are also used to build certificates.
D	Symmetric cypher needs two complementary and distinct keys, one for encrypting and another for decrypting. False. Symmetric cypher only uses a shared key in both algorithms.
E	All the above.
F	None of the above.

32. Cryptographic protocols. Key distribution:

A	With symmetric cypher, we only need to distribute the private key. False. We need to distribute a shared key.
B	With asymmetric cypher, we only need to distribute the public key. True. The private key is kept secret by each principal. So, it isn't distributed. Only the public key is propagated to other agents.
C	In symmetric cypher, information leakage is not a problem. False. In the key distribution procedure any information leakage is a critical problem when we use symmetric cypher. If unauthorised users get a copy of the shared key, they might gather all the information exchanged using that key.
D	In asymmetric cypher, we need secret channels to distribute keys. False. In asymmetric cypher only the public key is distributed. To this end, no secret channel is needed.
E	All the above.
F	None of the above.

33. In the basic (synchronous) request/reply pattern:

A	Client requests may be delivered in non-FIFO order to the server. False. The basic pattern is synchronous. This means that the client remains logically blocked until it gets the reply of the current request. As a result of this, it is impossible to break a FIFO order: FIFO refers to what each client does in its communication with the server, and a client cannot have more than a pending request. So, a new request cannot be processed before any of the previous ones.

B	The client can send another request before obtaining the reply for the current one. False. This is impossible in the basic synchronous pattern.
C	If the server crashes before replying a given request to the client, the client blocks forever. True. This was already shown in the activities from Seminar 4.
D	This pattern is implemented using PUSH and PULL ZeroMQ sockets. False. It is implemented using REQ and REP sockets.
E	All the above.
F	None of the above.

34. In order to deploy the basic request/reply pattern, considering ZeroMQ sockets...:

A	Clients bind their addresses and servers connect to them. False. Although this is conceptually possible, it only works when a given server always works with the same client. That is not the general case: a given server should be able to interact with as many clients as possible. To this end, the server should bind its address and clients should connect to it.
B	Servers bind their addresses and clients connect to them. True. Already explained in the previous sentence.
C	Both clients and servers need a single socket. Both client and servers bind their addresses and may also connect to other sockets. False. See the explanation given in the first item of this question.
D	No socket is needed to implement this architectural pattern. False. All architectural patterns need some kind of communication among their agents. To this end, sockets are needed.
E	All the above.
F	None of the above.

35. The basic PUSH-PULL architectural pattern...

A	...is a bidirectional communication pattern. False. It is unidirectional (from PUSH to PULL).
B	...is a synchronous communication pattern. False. It is asynchronous.
C	...is a multicast communication pattern. False. It implements point-to-point communication.
D	...is an asynchronous communication pattern. True.
E	All the above.
F	None of the above.

36. In the advanced client/server architectural pattern with multiple clients and multiple servers interconnected by an intermediate queue…

A	Each server instance is a single point of failure for that service. False. If one of the instances fails, there might be others that remain available and that may receive the forwarded requests. So, the failure of a server instance is not critical.
[✓ B]	The intermediate queue is stable. In its simplest configuration, its sockets are bound. True. The intermediate queue is the stable element in this architectural pattern. The easiest configuration consists in binding its two sockets to static addresses.
C	The intermediate queue uses PUSH-PULL sockets. False. The intermediate queue uses two ROUTER sockets in its regular configuration.
D	Clients use SUB sockets. False. Clients use REQ sockets in this architectural pattern.
E	All the above.
F	None of the above.

37. Heart-beats are used in some advanced client-server architectures in order to…

A	...detect client failures. False. They are used either for detecting server instance failures or to detect broker failures.
B	...monitor the current workload. False. The current workload does not need heart-beats in order to be reported to the broker. It may be piggybacked in regular replies or communicated using explicit messages or approximated considering the existence of pending replies from each server instance.
C	...improve the scalability of servers. False. Heart-beats require some communication effort. So, they do not introduce any scalability improvement in this architectural pattern.
D	...implement cryptographic protocols. False. Heart-beats are only needed for detecting process failures. They aren't related with security protocols.
E	All the above.
[✓ F]	None of the above.

38. Retries are used in some advanced client-server architectures in order to...

[✓ A]	...detect server failures. True. They are considered in multi-client/single-server configurations in order to detect server failures. They have also been considered in multi-client/multi-server with an intermediate queue in order to detect broker failures.

B	...implement a load balancing mechanism. False. Retries are needed for diagnosing a process failure, but not for balancing the load.
C	...improve the scalability of servers. False. As in question 37, any failure detection mechanism demands some effort. This doesn't boost performance nor scalability, but it is a requirement to ensure service availability.
D	...detect client failures, combining them with timeouts. False. Client failures are not a problem in the general case. Servers do not monitor client liveness in the architectural patterns explained in Unit 9.
E	All the above.
F	None of the above.

39. In order to implement a mechanism for recovering in case of a non-idempotent request failure, we need:

A	To identify clearly each request message, with an <id-sender, id-request> pair.
B	Before serving each request, the server should look for that request in its "reply store".
C	If a request is found in the "reply store", the reply message is taken from there and sent to the client.
D	After serving each request, the server copies the reply message to a local "reply store".
[✓ E]	All the above. These four statements describe the behaviour of the "reply store" mechanism that provides a general solution for the non-idempotent request failure problem.
F	None of the above.

40. In the advanced client/server architectural patterns from Unit 9, servers are replicated in the following way…

A	The active replication model is used. False. At the end of the unit, only a passive replication model implementation is described.
B	All service operations are idempotent. False. Replication does not demand that all operations were idempotent.
[✓ C]	The passive replication model is used. True. This unit describes in detail a passive replication model configuration with a primary and a backup replica.
D	Both primary and back-up replicas use a heart-beat module to detect client failures. False. Client liveness monitoring isn't needed in the advanced architectural patterns discussed in Unit 9.
E	All the above.
F	None of the above.