GROUP:		**Grade**	**C**	**I**	**N**
NAME:		SURNAME:			
SIGNATURE:		ID# (DNI):			

This exam consists of 40 multiple choice questions. In every case only one of the answers is the correct one. You should indicate your answer by writing an "X" within the corresponding cell to the left. 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 the correct value, which is -0,05 points. So, think carefully your answers.

If you have reasonable doubts, write an "*" (total doubt) or a number "3", at the end of the question's statement, and use the margins to explain further. The explanation must be brief.

1. Distributed systems...

	...always consist of a set of concurrent agents that can be run in a set of interconnected computers.
	...need to provide some degree of fault tolerance.
	...allow resource sharing.
	...may rely on message passing as their inter-agent communication mechanism.
	All the above.
	None of the above.

2. Some of the application areas in distributed computing are...

	...location, replication, migration, persistency, transactional, access, and failure transparencies.
	...WWW, sensor networks, *Internet of Things*, cooperative computing, highly-available clusters, cloud computing, etc.
	...producer-consumer with bounded buffers, readers-writer problem, dining philosophers problem, etc.
	...critical sections, distributed consensus, atomic broadcast, group membership, eventual consistency, 2-phase-commit protocol, nested transactions, etc.
	All the above.
	None of the above.

3. **The generic scientific-technical goal of cloud computing is...**

	...to design scalable algorithms.
	...making money.
	...to deploy fault-tolerant containers.
	...to create and exploit software services in a simple and efficient way.
	All the above.
	None of the above.

4. **In the cloud computing area there are several roles related to the software service life cycle. Those roles are...**

	Web, worker and VM.
	Monitoring, analysis, planning, execution and knowledge (MAPE-K).
	User, developer, administrator and provider.
	SaaS, PaaS and IaaS.
	All the above.
	None of the above.

5. **What is the relation between concurrent and distributed systems?**

	Every distributed system is also a concurrent system.
	Concurrent systems are not distributed.
	Distributed systems are not concurrent.
	Every concurrent system is also a distributed system.
	All the above.
	None of the above.

6. In a distributed system, its agents may interact…

	…using a message-passing mechanism.
	…following a client-server approach.
	…following a peer-to-peer approach.
	…sharing memory.
	All the above.
	None of the above.

7. In a distributed system…

	…each of its agents has a private state and does not interact with other agents.
	…agents may have their own state, but they collaborate in order to achieve a global goal.
	…agents are independent and they do not share resources.
	…concurrency is the source of many problems. So, modern distributed systems are not concurrent.
	All the above.
	None of the above.

8. The peer-to-peer interaction approach…

	…is not used in distributed systems.
	…assumes that agents are interested in some kind of resource and as soon as some agent B has an instance or fragment of such resource, B may distribute it.
	…clearly distinguishes between client agents and server agents.
	…is a strongly centralised interaction approach.
	All the above.
	None of the above.

9. The world-wide web…

	…is an example of distributed application following the peer-to-peer interaction approach.
	…uses web browsers as a particular type of server agent.
	…is a type of distributed application area where documents are transferred between servers and clients.
	…does not allow a client-server interaction approach.
	All the above.
	None of the above.

10. About the service models in cloud computing:

	IaaS: Provides general applications as its service. An example is Google Docs / Google Drive.
	SaaS: Automatizes application deployment and application elasticity. An example is Windows Azure.
	PaaS: Provides a virtual infrastructure as its service, where components can be deployed in a non-automatized way. Example: Amazon EC2.
	IaaS relies on the service provided by SaaS which also relies on the service provided by PaaS, defining a three-layered logical tree of service dependencies.
	All the above.
	None of the above.

11. The properties being required to distributed systems are...

	Centralised control.
	Daily software upgrades.
	Extremely high degrees of concurrency in each implemented agent.
	To be programmed in Node.js.
	All the above.
	None of the above.

12. Some of the fundamental problems (and their solutions) in distributed computing are...

	Component coordination (via message passing, designing algorithms that require a minimal message exchange).
	Failure management (using replication, failure detectors and recovery mechanisms).
	State persistence (via distributed commit protocols, persistent storage and replication).
	State consistency (using replication and consistency protocols).
	All the above.
	None of the above.

13. The distributed system model presented in Unit 2...

	...considers all low-level details about system behaviour. This guarantees a more precise result in the software design stage.
	...assumes that all agents are multi-threaded processes.
	...always assumes synchronous processes and synchronous communication.
	...represents the execution of processes as a sequence of interruptible actions or events.
	All the above.
	None of the above.

14. When we compare asynchronous servers with multi-threaded servers...

	Asynchronous servers implement in a trivial way the atomic actions defined in the proposed distributed system model in Unit 2.
	“Event-driven” is a synonym for “multi-threaded”.
	Asynchronous servers are commonly blocked due to concurrency while multi-threaded servers tolerate concurrent accesses to resources without blocking.
	JavaScript is an example of programming language specifically tailored to implement multi-threaded servers.
	All the above.
	None of the above.

15. Properties required to distributed systems…

	Fault-tolerance.
	High availability.
	Security.
	Scalability.
	All the above.
	None of the above.

16. State consistency means that…

	All the state being managed in a component may only have a single copy in all the system; e.g., it is stored in a centralised database.
	All global variables should be accessed in mutual exclusion in order to avoid race conditions.
	When a component is replicated, there is a set of invariants that limit the degree of divergence among the replicas of a specific data element.
	When a component is replicated, either all replicas are alive and work correctly or all they fail and are unable to work.
	All the above.
	None of the above.

17. State persistence means that…

	A distributed application cannot have volatile data. All managed data elements should have a copy in disk files or databases.
	The access to any data element should be always done in the context of a distributed transaction.
	Once a persistent data change is applied, its endurance should be guaranteed.
	Every secondary storage device being used by a distributed application should be replicated.
	All the above.
	None of the above.

18. In the distributed system model of Unit 2…

	Internal events refer to actions applied by the logic of the agent. For instance, to receive a message.
	Internal and external events generate state transitions.
	The execution of an agent is modelled as a sequence of events. So, it is always sequential and both multi-threading and concurrency cannot be represented.
	Since all distributed systems should be failure transparent, this model assumes that failures never happen.
	All the above.
	None of the above.

19. Communication in the simple system model of Unit 2…

	…assumes that internal events define a “locally-precedes” total-order relation in each agent.
	…assumes that external events define a “directly-causes” relation where an output event is the cause of an input event.
	The transitive closure of the “locally-precedes” and “directly-causes” relations defines the “causal” communication relation.
	The “causal” communication relation allows to identify unrelated events as “concurrent”.
	All the above.
	None of the above.

20. In order to specify programmes in the simple system model of Unit 2…

	The model assumes atomic guards, protected by actions.
	Atomic actions are a potential source of errors. Because of this, they are implemented as interruptible blocks of code in all programming languages.
	Guards are a potential source of race conditions. So, they are not used in multi-threaded programming languages.
	The model assumes atomic actions, protected by conditions (also known as guards).
	All the above.
	None of the above.

21. Middleware is a software layer that...

	...is placed between the hardware and the operating system.
	...guarantees failure transparency to the components of distributed applications.
	...relies on containers for deploying distributed services.
	...is implemented in JavaScript.
	All the above.
	None of the above.

22. Some characteristics of all middleware layers are...

	They provide standard APIs.
	They use standard interaction protocols.
	They provide services of general interest.
	They guarantee the interoperability of components deployed on distinct platforms.
	All the above.
	None of the above.

23. Distributed object systems...

	...need a middleware for managing remote object invocation.
	...are inherently less scalable than distributed systems based on message-oriented middleware.
	...have a higher coupling than non-object-oriented distributed systems.
	...are, in the common case, location-transparent.
	All the above.
	None of the above.

24. Message-oriented middleware...

	...is persistent when the sender remains blocked waiting for some kind of reply from the receiver.
	...is transient when the communication is managed by a broker agent.
	...may be persistent and broker-based.
	...may be synchronous and transient. ZeroMQ is an example of this type.
	All the above.
	None of the above.

25. Standards…

	…make interoperability easier.
	…cannot be used in distributed services.
	…guarantee failure transparency.
	…improve throughput.
	All the above.
	None of the above.

26. From a programmer’s point of view, when a standard is followed…

	…the programmes are easy to write, since there is a lower complexity in the handled elements.
	…the final result is more reliable, since the standard introduces clearly defined ways of doing things.
	…the obtained code has an easy maintenance since, although standards also change, those changes usually guarantee backwards compatibility.
	…the programmes are easy to write, since standards are based on well-defined and high-level concepts.
	All the above.
	None of the above.

27. Two approaches of remote method invocation in the web services area are…

	SOAP and REST.
	ZeroMQ and nanomsg.
	RPC and RMI.
	Client-server and peer-to-peer.
	All the above.
	None of the above.

28. The REST architectural style…

	Uses HTTP as its “transport”.
	Uses only four basic “methods”: GET, PUT, POST and DELETE.
	Uses its GET method for read-only actions.
	Takes the client-server architectural style as its basis, and promotes the use of stateless servers (in order to easily overcome failures).
	All the above.
	None of the above.

29. Some examples of “other middleware” are…

	SaaS.
	OAuth.
	Linux.
	MS-DOS.
	All the above.
	None of the above.

30. **Naming middleware…**

	…ensures failure transparency.
	…provides location transparency.
	…implements stateless servers.
	…improves system scalability.
	All the above.
	None of the above.

31. **A *Service Level Agreement* (SLA) is…**

	…a "contract" (i.e., an agreement) between a service provider and a service customer.
	…a specification of service characteristics (e.g., functionality, throughput, response time, availability…) and their levels to be guaranteed.
	…one of the aspects to be considered for deciding the number of instances of each service component at deployment time.
	…something to be considered in PaaS systems for filling the deployment and scaling plans for a given service.
	All the above.
	None of the above.

32. **In Unit 4, a service is…**

	…a distributed application that has been deployed and is active.
	…a set of independent scripts with a deployment plan.
	…a future distributed application that is still in its analysis or design stages.
	…a specific Node.js programme that is executed by a single user.
	All the above.
	None of the above.

33. These are some tasks to be considered when a distributed application is being deployed…

	To decide how many instances of each component should be run, and where.
	To decide which dependent services should be used by the distributed application being deployed.
	To decide the order in which each service component should be started.
	To contact the OS or container in each host to start its components.
	All the above.
	None of the above.

34. Service life cycle management is strongly related to deployment. Some of its tasks are…

	Component upgrades.
	Configuration changes.
	Component failure detection and recovery.
	Scale-out or scale-in decisions, depending on the current workload.
	All the above.
	None of the above.

35. Some problems that arise when a regular application is deployed in a desktop computer are…

	…software dependency resolution; i.e., to find the appropriate libraries such application depends on.
	…to give appropriate values to the environment variables being used by such application, if any.
	…to appropriately configure the application (e.g., via registry in Windows, configuration files in Linux, `/Library` files in Mac OS, etc.)
	…to find out if the application requirements are met by the current state of the target computer and operating system.
	All the above.
	None of the above.

36. Some of the elements in a deployment descriptor are…

	Naming middleware.
	Client command (e.g., docker).
	Filled deployment plan.
	Dockerfile.
	All the above.
	None of the above.

37. A component needs the following elements in order to be deployed…

	Its programme (or BLOB).
	A filled configuration template.
	A description of all its dependences.
	A specification of its endpoint.
	All the above.
	None of the above.

38. Dependency injection…

	…decouples the component code from any concrete implementation of dependences, and is supported by container environments.
	…requires the use of environment variables to resolve dependences.
	…requires the use of configuration files for resolving dependences.
	…solves all dependences statically at implementation time.
	All the above.
	None of the above.

39. In the IaaS service model…

	…deployment is completely automatized by the cloud provider.
	…several initial deployment decisions aren’t automatized: amount of component instances, type of VM required by each component…
	…life-cycle related deployment decisions are automatized; e.g., which workload levels throw scaling out/in actions, how to upgrade component SW…
	…no deployment support is given by the provider.
	All the above.
	None of the above.

40. In Windows Azure, some aspects of its deployment support are…

	There is a basic service upgrading plan, although it doesn’t support stateful services.
	Components are known as “roles”.
	There is a basic fault domain management that enhances service availability.
	There is no deployment sequencing plan.
	All the above.
	None of the above.