A

This exam consists of 20 multiple choice questions. In every case only one of the answers is correct. You must answer in a separate sheet. If correctly answered, each question contributes 0,5 points to your grade. If the answer is wrong, the contribution is negative: -0,167 points. In the answer sheet, fill carefully your chosen slot. To this end, use a dark pen or pencil.

THEORY

1. In the deployment plan of a distributed application...

a	The plan template states how to connect the components, listing both the dependencies to be resolved and the exposed endpoints. *True. Among other things the deployment plan template must state those aspects.*
b	The configuration templates for each component maintain the same values for every component instance. *False. A configuration template should not declare the final values for the parameters it considers in order to deploy a component instance. A template should only provide the set of parameters to be defined and when an administrator (or deployer) fills that template he or she will assign the appropriate values to those parameters. In the common case different instances will have different values, at least for some of those parameters.*
c	Several instances of a component may have unresolved dependencies at deployment time. They will be resolved later, while the service is running. *False. In the regular case, component instance dependencies must be resolved at deployment time. It is dangerous to try to solve unbound dependencies when the service is already running since, if any problem is found, many ongoing requests might be lost and several service components would not start or run correctly.*
d	The plan template states the location (nodes) for every component instance respecting the constraint that, at each node, one and only one instance of each component must be run. *False. A template is still open in this regard. Once the template is filled no constraint requires that each component instance be deployed in a different node or that every node must only host a unique component instance of a given service.*

A

2. Considering failure models for distributed systems...

a	In both the ***stop*** and ***crash*** failure models, the failure of a process may always be detected by other processes. *False. Process failures may only be safely detected by other processes in the stop failure model. That feature is not assumed in the crash failure model. Note that the stop failure model might assume a higher degree of synchrony than the crash model in order to make possible that stop failures were detected.*
b	It is easy to implement a distributed system whose failures match the ***stop*** or ***crash*** failure models. *False. Indeed, it is very difficult that a system may ensure that all its processes behave perfectly until they stop or crash. To this end we would need perfect programmes (without any bug and with a configuration that perfectly matches their current real environment) in all processes that run on that system, besides a perfect set of components in the system itself (i.e., operating system being run in each node, middleware being used, etc.). It is difficult to achieve that degree of quality in software development and deployment tasks.*
c	A ***network partition*** model is equivalent to the ***general omission*** (i.e., send-omission and receive-omission) failure model. *False. A network partition avoids that processes placed in different network parts could exchange messages, but the processes that are placed in the same part could communicate among them without any problem. On the other hand, the general omission failure model assumes that processes may find problems when they try to send or receive messages, independently of the (respectively) receiver or sender of those messages. As we can see, both concepts are not equivalent.*
d	When failure transparency is ensured assuming a ***Byzantine*** failure model, then it is also ensured for all the remaining failure models. *True. If we can provide failure transparency assuming the Byzantine (or arbitrary) failure model, this implies that every kind of failure may be overcome by our software. Note that the Byzantine model subsumes all the other models. Because of this, the failures being assumed in the remaining failure models are already considered in the Byzantine model and they are already overcome when failure transparency is reached in the arbitrary failure model.*

3. Regarding replication models...

a	The passive replication model supports the ***Byzantine*** failure model but it does not support the ***crash and link*** failure model. *False. In order to support the Byzantine failure model we need to know how many simultaneous arbitrary failures may happen at a time. Let assume that "f" is that value. Once this value is known we need at least 2f+1 active replicas in order to overcome arbitrary failures in the best possible scenario or 3f+1 active replicas in case of using messages whose sender cannot be authenticated. We consider that an active replica is one that may directly interact with the service clients. In the passive replication model only an active replica exists per replicated agent: its primary replica. Because of this, the passive replication model cannot overcome any scenario with arbitrary failures.* *Another argument for considering that this sentence is false consists in noting that "crash and link" failures are more benign (i.e., easier to manage) than "Byzantine" failures. Due to this, if any software may assume and properly manage (i.e., overcome) Byzantine failures then it will certainly overcome crash and link failures, too.*

b	In the passive replication model, update propagation in total order is very difficult to implement. *False. Update propagation in total order is very easy to implement in that model. Since there is a single update propagator (the primary replica), it only needs to number and sequence the update messages that it propagates. It only needs a local counter to this end.*
c	In the active replication model, reconfiguration after a replica failure is a very difficult procedure. *False. Since all replicas are identical in that model, in the regular case no reconfiguration is needed when one of them fails.*
d	The active replication model demands more replicas for supporting ***Byzantine*** failures than for supporting ***crash*** failures. *True. In the active replication model both types of failures may be supported. In order to support "f" simultaneous crash failures only f+1 replicas are needed but when we need to support "f" simultaneous Byzantine failures either 2f+1 or 3f+1 replicas will be needed, depending on the possibility of authenticating the message senders.*

4. NoSQL datastores are easily scalable because...

a	They guarantee referential integrity. *False. Many NoSQL datastore do not guarantee referential integrity.*
b	They reduce the redundancy of the stored data. *False. In order to reduce data redundancy the normalisation techniques were introduced in relational databases. However, NoSQL datastores are not based on normalisation in order to define their database schemas.*
c	They simplify or eliminate transactions. *True. ACID transactions (that are used in relational DBMSs) have implicit concurrency control mechanisms that may block running transactions. Any kind of agent synchronisation may endanger service scalability. Many NoSQL datastores have eliminated transaction management from their provided set of services. Because of this, they have renounced to isolation and atomicity in favour of performance and scalability.*
d	They guarantee that network partitions never happen. *False. Network partitions occur due to problems in the configuration or liveness of the communication hardware (routers, switches, etc.). NoSQL datastores do not have any direct effect on those aspects.*

5. According to the CAP theorem, it is possible that when a network partition arises...

a	...all system subgroups go on, answering client requests, and replicated services will ensure at least sequential consistency. *False. When a network partition occurs, the processes that belong to different parts of the network cannot communicate exchanging messages. In that case, if all parts go on, the updates applied in one of them cannot be propagated to any other. Because of this, the processes in different network parts cannot maintain nor see the same sequence of updates on their shared variables. As a result of this, the consistency maintained in that system cannot be sequential.*

b	...a partitionable model is assumed and replicated services ensure eventual consistency. *True. If a partitionable model is assumed the processes located in every network part may go on. As a result of this, they lose strong consistency but the resulting consistency can still respect the "eventual consistency" specification. This means that, once the network connectivity is recovered, the replicas will exchange their applied updates and all they will converge to the same state (i.e., they will have the same values in each shared variable).*
c	...the primary component model must be assumed and every service replica must answer client requests. *False. There is a contradiction in this sentence. If the primary component model is assumed then only the network part or subgroup with more than a half of the service replicas will be able to continue. In that case, it is not true that "every service replica answers its client requests" since in minor subgroups that condition is not respected.*
d	...all service replicas must be stopped until the network connectivity is repaired. No service activity is tolerated in that interval. *False. As it has been explained in part "b", it is not mandatory in all cases that the service replicas remain stopped while the network is partitioned.*

6. Contention (i.e., performance bottlenecks that compromise service scalability) may be caused by...

a	The usage of decentralised algorithms for managing heavy tasks. *False. If a long task is managed with a decentralised algorithm then it is divided in multiple subtasks and each subtask is run by a different agent. In that case, that task is completed soon. As a result of this, no contention is introduced by this approach.*
b	A wrong distribution of the resources that originates dense network traffic. *True. If the resources needed for managing any distributed tasks are distributed in an inappropriate way, the agents that are collaborating in that execution will need to exchange many additional messages and those messages may introduce unwanted synchronisation among those agents. As a result, the participating agents may be blocked for long intervals waiting for some expected messages and this increases the contention, reducing service scalability.*
c	The usage of an asynchronous communication middleware. *False. Inter-agent communication must be as asynchronous as possible with the goal of minimising any blocking interval.*
d	The replication of the components that balance the load among the worker processes. *False. If a component is replicated with care it may increase its performance and reduce its blocking intervals.*

7. General principles for achieving scalability:

a	To increase the concurrency degree without bound. *False. An increase in the concurrency degree does not necessarily improve the scalability of a service. If the different concurrent activities need to access any shared resource they will need some concurrency control mechanisms and this may lead to their blocking. In that case, increasing the concurrency degree might reduce the scalability of that service.*

b	To reach a strong consistency. *False. The stronger a consistency is, the longer its blocking intervals (needed to achieve that level of consistency) will be. Therefore, in order to improve scalability we need weak consistency models.*
c	To avoid inter-agent synchronisation as much as possible. *True. The key condition for improving scalability is to reduce inter-agent synchronisation to a minimum. In this way, agents will invest as much time as possible in executing their assigned tasks instead of remaining blocked due to synchronisation steps.*
d	To save all service data in secondary storage in order to ensure its persistency. *False. Saving data in secondary storage introduces long I/O intervals. The agents that request those operations remain suspended in those intervals. Therefore, this doesn't improve scalability.*

8. Peter is a security expert that works in company B. He used yesterday a packet sniffer and obtained the ID and password of a system administrator from company A. He also found the public address of one of the A's servers where company administrators may remotely control the other servers in the company. For company A, the current scenario is an example of...

a	...a denial of service attack. *False. At the moment the service capacity of company A has not received yet any attack that might reduce or compromise it.*
b	...a potential external threat. *True. It is a threat, since some security mechanisms being used in A are now known by people that do not belong to the company, but such knowledge has not been translated into any action that may damage A. It is external since Peter does not belong to A.*
c	...a weakness in the routing protocols. *False. Routing protocols have no direct specific effect on how packet sniffers work. In order to avoid any threats introduced by packet sniffers we may use encrypted communication and encrypted communication does not depend on any routing protocol. Encrypted communication may be implemented at the application layer while routing tasks are managed at the network layer.*
d	...a physical security mechanism. *False. What has been described here is a security threat, not a security mechanism.*

SEMINARS

9. Given this sequence of Docker commands executed from the CLI:

```
docker pull fedora
docker run --name fedora fedora dnf install –y nodejs
docker commit fedora node
docker push node
```

Which of the following actions hasn't been done?

a	Download an image (*fedora*) from the public repository (i.e., from the Docker Hub). *False. This has been done in the "docker pull fedora" command.*
b	Upload an image (*node*) to the Docker Hub. *False. This has been done in the "docker push node" command.*
c	Create a container and run the *node* command in it. *True. The container has been created using the "docker run" command, but in that line the user hasn't requested the execution of the "node" command in that container.*
d	Create a container, modify it and create a new image from that container. *False. The container has been created in the "docker run" line. In that same line we have modified its contents using the "dnf install –y nodejs" Fedora command. Later on, the "docker commit" line has created a new image from that modified container.*

10. Considering this Dockerfile:

```
FROM zmq
RUN mkdir /zmq
COPY ./worker.js /zmq/worker.js
WORKDIR /zmq
CMD node worker $BROKER_PORT_8001_TCP
```

If we generate a Docker image from it using the following command:

```
docker build –t worker .
```

Which is the FALSE sentence?

a	The *worker* image is a modification of the *zmq* image. *True. The "docker build" command has created a "worker" image using the Dockerfile presented in this question. The first line from that Dockerfile states that such new image takes the existing "zmq" image as its base and both the "RUN" and "COPY" instructions have extended and modified such "zmq" image.*
b	The working directory for the CMD command is */zmq*. *True. That is the effect of the WORKDIR line in the Dockerfile.*
c	If a container is created from the *worker* image, it will run the programme cited in the CMD command. *True. That is the effect of the CMD line in the Dockerfile.*
d	If a container is created from the *worker* image, it won't run any programme since programmes should be specified using ENTRYPOINT instead of CMD. *False. Both ENTRYPOINT and CMD may be used for specifying the programme to be run in the container.*

A

11. Assume that we have a default Docker installation in our computer where we have created a "node2" image able to run the node interpreter from the command line (i.e., "node2" is an image, not a container). Imagine that we want to execute a node.js programme that we have in our computer (e.g., the file "/tmp/example.js") in a Docker container. To this end, among other actions, we should...

a	Use **docker run node2** from the command line, passing the programme pathname as its last argument; i.e. **docker run node2 /tmp/example.js** *False. In the "docker run" command the first argument refers to the image name and the second, when given, refers to the command to be run inside that container. That second argument cannot be the pathname of a file maintained in the host. Instead of this, it should be something placed in the container.*
b	Nothing can be done since the files in the host computer cannot be used by a running container and there is no way to transfer those files to an image. *False. There are several ways of transferring host resources to the image.*
c	Copy that file into a new image based on the **node2** one. To this end we may use the COPY command in a Dockerfile. *True. This has been the mechanism explained in the examples shown in Seminar 4.*
d	Use **docker cp /tmp/example.js node2**. *False. There is a "docker cp" command but is uses a different syntax for copying files into an image. The correct syntax is:* **docker cp /tmp/example.js node2:** *Optionally, after the colon (":") that follows the image name we may have specified a pathname. The colon symbol is mandatory. It is needed for specifying which argument refers to the image. The other argument is a valid pathname in the host computer.*

12. Considering this Dockerfile...

```
FROM fedora
RUN dnf install -y nodejs
RUN dnf install -y zeromq-devel
RUN dnf install -y npm
RUN dnf install -y make
RUN npm install zmq
```

The following sentences are true:

a	This Dockerfile doesn't make sense since it has no ENTRYPOINT or CMD commands in it. It does nothing at all. *False. A Dockerfile like this already makes sense. It extends the default "fedora" image installing additional packages in the resulting image. ENTRYPOINT and CMD commands are not mandatory in Dockerfiles.*

b	This Dockerfile doesn't work since it fails in its second line. There is no "dnf" instruction in Docker. *False. The "dnf" command is not a Docker-related command but a Fedora command. It may be used without problems in a RUN Docker command when the image being taken as a base uses a Fedora operating system.*
c	The name of the image being created with this Dockerfile is "zmq". *False. The name to be given to the resulting image must be specified using the "docker build" command.*
d	This Dockerfile creates a new image based on the "fedora" one. The new image has added at least 4 fedora packages onto the base "fedora" image. *True. Those are the actions applied in this example. Besides this, this Dockerfile also runs the "npm" command in order to install the Node.js "zmq" module in the generated image.*

13. The implementation of a weak consistency model (Activity 1 from Seminar 5, source files: shared1.js and proc1.js) using a PUB/SUB communication pattern guarantees the following consistency model:

a	Sequential since the writes of every process are immediately forwarded to other processes using a multicast from the PUB socket. *False. Since there may be many writer processes and each of them uses a different PUB socket, their written values being propagated may arrive in a different order to each other process. As a result of this, the system processes do not see an agreed sequential order of values and the resulting consistency is not sequential.*
b	Causal since every process reads the values written by the remaining processes before starting its own writes on the shared variable. *False. Since messages need some time to be propagated nothing guarantees that a writer is able to read the values previously written by other processes before the propagation of its own written value is started. Therefore what is mentioned in this sentence cannot be guaranteed.*
c	Cache since the subscription to the writes on each variable guarantees that all processes read the same sequence of values from each variable. *False. Since each process uses a different PUB socket and every process might write new values onto the same variable, there is no guarantee that those writes are received in the same order by all processes. Therefore, cache consistency is not ensured.*
d	FIFO since each process propagates its writes in order using its PUB socket and the protocol being used (TCP) respects FIFO order. *True. PUB sockets usually respect FIFO order. Since every process has its own PUB socket, this means that the values written by each process are seen by the remaining processes in the same writing order. This matches the conditions required by the FIFO consistency model.*

A

14. Considering the implementation of a replication protocol based on a sequencer process, as that described in the bulletin from Seminar 5...

a	When we run the sequencer process in the fastest node of our system, the resulting consistency model is fast. *False. A consistency model is fast when it doesn't require any message propagation in order to consider that a local read or write operation has been completed. The usage of a sequencer process implies that a writer needs to forward a message to the sequencer before considering that such write action has terminated. Because of this, no sequencer-based consistency model may be fast.*
b	If we use a different sequencer process for each variable, we will provide cache consistency without ensuring sequential consistency. *True. With a different sequencer per variable we may ensure that all processes agree on the order of values per variable. This is the condition being required by the cache consistency model. The sequential model, on the other hand, requires agreement on a common sequence for all processes and all variables. This cannot be ensured using more than one sequencer.*
c	If we use the same sequencer process for all variables, we will provide sequential consistency without ensuring cache consistency. *False. Since the sequential consistency model is stricter than the cache consistency model, when a protocol implements sequential consistency it necessarily implements cache consistency, too.*
d	If we use the same sequencer process for all variables, we will provide sequential consistency without ensuring FIFO consistency. *False. Since the sequential consistency model is stricter than the FIFO consistency model, when a protocol implements sequential consistency it necessarily implements FIFO consistency, too.*

15. Considering this node.js programme...

```
var cluster = require('cluster');
var http = require('http');
var numCPUs = require('os').cpus().length;
if (cluster.isMaster) {
  for (var i=0; i < numCPUs; i++) cluster.fork();
  cluster.on('exit', function(who, code, signal) {
    console.log('Process '+ who.process.pid +' died');
  });
} else {
  http.createServer(function(req, res) {
    res.writeHead(200);
    res.end('hello world\n');
  }).listen(8000);
}
```

The following sentences are true:

a	This programme fails when the second worker is created since all workers are trying to use the same port (8000) and that port is already in use. *False. When we use the "cluster" module in order to create workers using the cluster.fork() function, all those workers may share a common set of ports. They will not generate any "port already in use" error.*

b	The master process successfully creates as many worker processes as CPUs (or cores) exist in the local computer. *True. This is the goal of the "for" loop seen in this example.*
c	The "cluster" module may deploy each generated worker process in a different computer. *False. The "cluster" module may only generate worker processes in the local computer.*
d	This programme prints a message when the master dies. *False. It prints a message each time a worker process dies but it does not print any thing when the master is terminated.*

16. Regarding the programme shown in question 9, the following sentences are true... [This was commented in the classroom where the exam took place: instead of "question 9", it should be "the previous question" (i.e., question 15]

a	The first worker prints a message each time it receives a message. *False. No message is printed when a request message is received. Instead of this, an HTTP response message is returned to the client.*
b	The answer being returned by the worker depends on the contents of the HTTP request sent by the client. *False. The HTTP response message being returned to clients is always the same ("hello world"), independently on the request being received.*
c	Master-worker communication has been implemented using a REQ/REP communication pattern. *False. There is no explicit master-worker communication in this example.*
d	Each worker is an HTTP server process. *True. The implementer of this example has used the http.createServer() function for building the code of each worker process. As a result of this, they are HTTP server processes.*

17. In order to enhance its scalability, MongoDB uses...

a	...the active replication model. *False. It uses a slight extension of the passive or primary-backup replication model in which backup replicas are able to serve read-only requests and there may be "virtual" processes participating in the internal voting procedure for choosing a new primary replica when the previous primary has failed.*
b	...a partitionable model for dealing with network partitions. *False. MongoDB uses a primary partition model in order to manage network partition failures.*
c	...database sharding, combined with primary-backup replication. *True. As we have already explained in part "a", MongoDB may use primary-backup replication. Additionally, when the database is large, MongoDB may distribute its contents onto multiple mongod servers using sharding.*
d	...transactions that respect the four ACID properties. *False. MongoDB does not use atomic transactions encompassing multiple database accessing statements.*

18. When a MongoDB database is partitioned, all its shards need to have a similar size. This is achieved using...

a	...“*journaling*”. *False. Journaling is used in MongoDB but with another goal. Journaling is needed for ensuring that any database update is durable. To this end, the access is first described in a log file and once it has been written in the log it is written in the appropriate database collection. If the database manager fails while this write is being done at least the log description would be written, allowing the update completion once the database manager is restarted.*
b	...a “chunk migration” algorithm. *True. This algorithm is needed for migrating chunks from larger shards to smaller shards in order to balance their size.*
c	...normalization. *False. Normalisation is not used in MongoDB. Normalisation is needed in relational databases in order to eliminate data redundancy as much as possible in the database schema.*
d	...a mutual exclusion algorithm. *False. Mutual exclusion algorithms are needed in concurrent programming in order to avoid race conditions. They do not have any direct relation with shard balancing in partitioned databases.*

19. Regarding the thematic classification of security vulnerabilities...

a	“Phishing” attacks exploit “social engineering” vulnerabilities (i.e., the exploited vulnerabilities belong to the “social engineering” class in this case). *True. A phishing attack consists in impersonating a well-known service in order to get the identity and passwords (or other relevant secret data) from users of that service. It exploits “social engineering” vulnerabilities.*
b	“Protocol design” vulnerabilities belong to the “social engineering” class. *False. Vulnerabilities in protocol designs belong to the “software error” class.*
c	“Internal espionage” vulnerabilities belong to the “software error” class. *False. Internal espionage vulnerabilities belong to the “social engineering” class.*
d	“Personal protection” faults belong to the “software error” class. *False. Personal protection vulnerabilities belong to the “faults in security policies” class.*

20. The origin-based classification of vulnerabilities...

a	...identifies four vulnerability classes: social engineering, software errors, faults in security policies and general weaknesses. *False. Those vulnerability classes are identified in the thematic classification of security vulnerabilities.*

A

b	...also considers the time needed to exploit vulnerabilities (and react to it in case of attack) and the degree of interaction demanded for that exploitation. *False. Those two dimensions are considered in the thematic classification of vulnerabilities.*
c	...considers that the origin of vulnerabilities will provide us with a guide to remedy them. *True. This is the main criterion or reason that leads to adopt that classification of vulnerabilities.*
d	...states that the attacks are the cause (i.e., the origin) of vulnerabilities. *False. In order to start a security attack, the attacker needs to identify some vulnerability in the system to be attacked. As a result, we have the opposite relation: without vulnerabilities there cannot be attacks. Therefore, vulnerabilities are the origin of (or, at least, what allows) attacks.*