Aquest examen consta de dues seccions de 20 qüestions de múltiple opció cadascuna. Cada secció correspon a cada parcial fet fins ara i serà avaluat independentment. En cada qüestió només hi ha una resposta correcta.

Dins de cada parcial totes les qüestions tenen el mateix valor. Si la resposta és correcta, aportarà 0.5 punts a la nota del parcial. Si la resposta és incorrecta, descomptarà 0.1 punts. Tinga-ho en compte a l'hora de contestar. En cas de dubte, deixe la qüestió en blanc.

Les respostes han de ser facilitades en una fulla SEPARADA que es facilitarà al costat d'aquest enunciat. En la fulla de respostes haurà d'indicar a quin parcial es presenta (o a tots dos), emplenant les caselles corresponents.

Cada parcial pot ser completat en menys d'una hora, però disposa de dues hores que podrà distribuir segons preferisca.

Primer parcial

1. Els sistemes distribuïts…

A	Han de programar-se sempre utilitzant llenguatges de programació asincrònica. FALS. No hi ha restriccionés sobre el llenguatge de programació. No s'exigeix programació asincrònica.
B	Han de desplegar-se sempre utilitzant més d'una màquina física. FALS. Ha d'existir més d'un component en l'aplicació distribuïda, però poden arribar a desplegar-se tots sobre una mateixa màquina física.
C	Han de compartir sempre un rellotge global entre tots els agents. FALS. Aquest nivell de sincronia entre tots els agents és poc recomanable ja que resulta extraordinàriament difícil garantir-ho en un sistema real.
D	Han d'utilitzar sempre RPC per a intercomunicar els agents. FALS. Existeixen altres tipus de mecanismes d'intercomunicació entre agents (en el tema 9 hem estudiats uns quants patrons arquitectònics, que precisament modelen el tipus d'intercomunicació). Res obliga al fet que tota la comunicació es realitze sota un patró arquitectònic petició/resposta.
E	Totes les anteriors.
F	Cap de les anteriors.

2. Sobre la computació en el núvol

A	És un exemple de sistema distribuït amb comunicació P2P. FALS. La computació en el núvol no exigeix (de fet, ni tan sols ho recomana) que la comunicació entre agents seguisca un model P2P. En una arquitectura P2P tots els agents exerceixen rols similars i no es pot parlar de clients ni servidors, sinó de "servents" (cada agent exerceix tots dos rols simultàniament). Per contra, la majoria dels sistemes de computació en el núvol ofereixen serveis als seus usuaris. Sí que és habitual distingir entre clients i servidors en aquest tipus de sistemes.
B	Millora l'escalabilitat dels sistemes distribuïts aplicant tècniques utilitzades prèviament en els "*mainframes*". FALS. La computació en el núvol necessita escalabilitat horitzontal. Les tècniques utilitzades en els "*mainframes*" mai van arribar a estar basades en aquest tipus d'escalabilitat.
C	La seua arquitectura típica consta d'un nivell inferior amb un servei IaaS, un nivell intermedi amb un servei PaaS i un nivell superior amb serveis SaaS. CERT. El nivell inferior facilita una infraestructura (IaaS), el nivell intermedi ofereix una plataforma (PaaS) i el nivell superior ofereix aplicacions com a servei (SaaS).
D	Està basada en l'ús d'estàndards SOAP. FALS. No es basa exclusivament en aquest estàndard. Els serveis proporcionats necessiten dependre d'aquest estàndard.
E	Totes les anteriors.
F	Cap de les anteriors.

3. L'objectiu global de la computació en el núvol és...

A	Forçar al fet que tots els usuaris reemplacen els seus ordinadors personals ("*desktops*") per clients lleugers. FALS. Els ordinadors personals poden ser utilitzats com a clients en aquests entorns, sense cap problema. Res força al fet que els usuaris els substituïsquen per un altre tipus d'ordinador.
B	Facilitar la creació d'aplicacions distribuïdes el comportament de les quals estiga afectat per fenòmens atmosfèrics. FALS. Els fenòmens atmosfèrics no tenen res a veure amb el comportament de cap aplicació distribuïda desenvolupada sobre aquests entorns.
C	Facilitar un entorn dels desenvolupadors per a depurar algorismes distribuïts. FALS. Pot ser que es facilite aquest tipus d'entorn als desenvolupadors d'aplicacions distribuïdes, però aquest no és l'objectiu global de la computació en el núvol.
D	Simplificar i incrementar l'eficiència en la creació i explotació de serveis de "*programari*". CERT. Aquest sí és l'objectiu principal: Facilitar la creació i desplegament de serveis, així com la gestió del seu cicle de vida, perquè resulte més senzill invertir en aquest tipus de serveis distribuïts.
E	Totes les anteriors.
F	Cap de les anteriors.

4. Normalment els proveïdors de computació en el núvol necessiten aquests mecanismes...

A	Tecnologies Java Virtual Machine. FALS. Res força al fet que s'utilitze Java (ni cap altre llenguatge o entorn de programació) per a implantar els serveis de computació en el núvol.
B	Tecnologies .NET Virtual Machine. FALS. Per les mateixes raons que en l'apartat anterior.
C	Tecnologies JavaScript Virtual Machine. FALS. Per les mateixes raons que en el primer apartat.
D	Tecnologies de virtualització de *maquinari*, basades en hipervisor. CERT. A l'hora de donar suport a la infraestructura se solen utilitzar tecnologies de virtualització dels equips (és a dir, del *maquinari*). D'aquesta manera és possible executar múltiples màquines virtuals en un mateix ordinador físic.
E	Totes les anteriors.
F	Cap de les anteriors.

5. Alguns dels problemes fonamentals en sistemes distribuïts són...

A	Gestionar les fallades de components. CERT. Tot component pot arribar a fallar en algun moment i l'aplicació en la qual s'utilitze ha de ser capaç de reaccionar davant tal situació i seguir disponible. Sol recórrer-se a la replicació de components per a resoldre aquests problemes.
B	Coordinar les accions de diferents components. CERT. Una aplicació (o sistema) distribuïda consta sempre de múltiples components que col·laboren entre si per a aconseguir un objectiu comú, oferint la imatge de sistema únic i coherent. La coordinació està implícita en aquesta col·laboració.
C	Facilitar persistència d'estat. CERT. Per a tolerar certes situacions de fallada resulta necessari persistir l'estat de cada component. Això pot fer-se de diverses maneres: mantenint-ho en emmagatzematge secundari, replicant-ho en múltiples instàncies amb diferents fonts possibles de fallades...
D	Garantir la consistència de l'estat del sistema. CERT. La imatge proporcionada per un sistema distribuït ha de seguir sent la d'un sistema únic i coherent. No és recomanable que la imatge percebuda en interactuar amb el sistema depenga del node utilitzat (i que en cada node hi haja inconsistències respecte a l'estat dels altres).
E	Totes les anteriors.
F	Cap de les anteriors.

6. Aquests són alguns dels problemes pràctics importants en sistemes distribuïts...

A	Garantir el compliment de les polítiques de seguretat. CERT. Aquest va ser l'objectiu descrit en el tema de seguretat. És un problema pràctic en qualsevol sistema distribuït.
B	Decidir l'arquitectura de les màquines físiques que executaran les aplicacions. FALS. L'arquitectura física dels ordinadors utilitzats no hauria de ser rellevant a l'hora de desenvolupar una aplicació distribuïda. Existeixen tant llenguatges de programació com "middleware" capaços de proporcionar independència de l'arquitectura subjacent.
C	Decidir la millor ubicació per als centres de processament de dades. FALS. Això pot resoldre-ho el proveïdor IaaS (en cas que s'arribe a requerir un IaaS) o qui haja de desplegar les aplicacions distribuïdes a utilitzar. No és un problema pràctic per a qui dissenye o utilitze un sistema distribuït. No sempre existirà la possibilitat de triar quina és la millor ubicació (per exemple, empreses xicotetes amb pressupost reduït que ja disposen del seu propi centre de dades i encara no consideren convenient la migració al model de computació en el núvol).
D	Trobar un bon proveïdor IaaS. FALS. No tots els sistemes distribuïts estan forçats a utilitzar un servei IaaS.
E	Totes les anteriors.
F	Cap de les anteriors.

7. Sobre el model simple de sistema distribuït explicat en les classes...

A	Un procés es modela com una seqüència d'esdeveniments executats atòmicament.
B	La relació *precedeix-localment* pot utilitzar-se per a ordenar indirectament dos esdeveniments de processos diferents.
C	Els esdeveniments de comunicació sempre són esdeveniments externs.
D	Tot esdeveniment de *recepció de missatge* té un esdeveniment d'enviament *de missatge* associat.
E	Totes les anteriors. Cadascuna de les afirmacions anteriors descriu una característica del model simple de sistema distribuït. Totes elles són certes.
F	Cap de les anteriors.

8. Sobre l'estat en un sistema distribuït...

A	Un procés A pot utilitzar l'estat d'un altre procés B per a decidir les seues pròpies transicions d'estat (és a dir, les de A). FALS. Un procés A no té manera directa de conèixer l'estat actual d'un altre procés B. Pot fer-ho sobre la base dels missatges que li envie B, però en aquest cas el contingut dels missatges pot dir-se que afecta al propi estat de A. Per tant, les decisions sempre haurien de prendre's sobre la base d'informació local.
B	Amb canals FIFO, tot procés pot observar el mateix valor d'una variable determinada. FALS. Si tots els processos pogueren observar simultàniament el mateix valor per a una variable concreta estaríem parlant d'un model de consistència estricta. La consistència estricta requereix protocols de consistència (i sincronització) complexos. Utilitzar canals FIFO no és suficient.
C	L'estat d'un sistema distribuït té una base local: les transicions per esdeveniments interns es decideixen únicament sobre la base de l'estat local del procés. CERT. Ja s'ha justificat en el primer apartat.
D	En un sistema distribuït tots els processos usen un mateix conjunt de variables. FALS. Normalment cada procés implanta un component diferent de certa aplicació distribuïda. Cada component pot manejar les seues pròpies variables, que no sempre dependran de les dels altres components.
E	Totes les anteriores.
F	Cap de les anteriors.

9. Objectius dels middleware...

A	Han de facilitar la interacció entre components desenvolupats independentment. CERT. Els middleware normalment faciliten la interacció entre components.
B	Han de facilitar la interacció entre components desplegats de manera autònoma. CERT. Els middleware normalment faciliten la interacció entre components, independentment de qui els desenvolupe o com es despleguen.
C	Evitar errors. CERT. Un middleware proporciona certa funcionalitat comuna a un bon nombre d'aplicacions distribuïdes. Se suposa que el middleware ha sigut suficientment provat i depurat abans de la seua utilització. Per això, evita que es donen errors a l'hora d'implantar aquesta funcionalitat (no és necessari escriure-la des de zero en les noves aplicacions; es reaprofita el que ja està en el middleware).
D	Reduir la complexitat en la implantació de serveis. CERT. Serveix l'explicat en l'apartat anterior.
E	Totes les anteriors.
F	Cap de les anteriors.

10. Els estàndards…

A	Introdueixen formes contrastades i verificades de fer les coses. CERT. Perquè un estàndard siga acceptat i aprofitat ha de proposar una solució ja contrastada per a cert problema. Diverses organitzacions han de discutir cada proposta d'estàndard fins que aquest arriba a ser acceptat. Durant aquest procés es comprova que la proposta tinga sentit i siga fàcil la seua adopció posterior per part d'altres empreses o organitzacions.
B	Són sempre generats per organitzacions especialitzades l'única funció de les quals és la publicació d'estàndards. FALS. En la redacció i publicació d'un estàndard solen participar representants d'un bon nombre d'empreses d'aquest sector.
C	Sempre especifiquen com implantar la funcionalitat que descriuen. FALS. Poden limitar-se a l'especificació de les interfícies a respectar o el protocol a utilitzar però no com s'implanten.
D	Implanten els middleware. FALS. Just al contrari. Els middleware implanten estàndards.
E	Totes les anteriors.
F	Cap de les anteriors.

11. Els middleware de nomenament...

A	Proporcionen transparència d'ubicació. CERT. Solen proporcionar aquest tipus de transparència ja que “tradueixen” un nom en una adreça o referència que no sempre proporciona una descripció exacta de la ubicació de l'entitat nomenada.
B	Han d'utilitzar codificació UTF-8 per a gestionar els noms. FALS. No exigeixen una codificació determinada per als noms.
C	Sempre es basen en xarxes IP. FALS. Els middleware solen estar sobre el nivell de transport. Un middleware de nomenament no és cap excepció. Per tant, en treballar per dalt del nivell de transport no depenen del tipus de xarxa que hi haja per baix. Poden utilitzar-se sobre qualsevol nivell de xarxa (el transport els aïlla d’aquests detalls); no s’exigeix que siguen xarxes IP.
D	Només proporcionen traduccions dels noms d'ordinador a les seues adreces IP. FALS. Aquest tipus de resolució es dóna en DNS, per exemple, però existeixen molts altres serveis de nomenament. Per exemple, en Java RMI (utilitzat en CSD el curs passat) el resultat d'una operació de resolució de nom era una referència a objecte (i no una adreça IP, que és el que implica aquesta afirmació).
E	Totes les anteriors.
F	Cap de les anteriors.

12. Sobre els sistemes d'objectes distribuïts...

A	Proporcionen transparència d'ubicació per als objectes del sistema. CERT. Solen utilitzar proxies i esquelets per a aconseguir la transparència d'ubicació. Això també es combina amb un servei de noms que permet obtenir les referències necessàries per a construir els proxies.
B	Faciliten l'intercanvi de referències a objecte entre els diferents processos. CERT. A més d'obtenir-les en la resolució de noms, també es reben com a arguments quan s'invoquen alguns mètodes.
C	Solen generar sistemes distribuïts amb un acoblament alt. CERT. No és rar que un objecte dispose de referències a altres objectes de l'aplicació i que arribe a utilitzar aquests objectes remots de manera transparent, augmentant l'acoblament entre components.
D	Inclouen a Java RMI com un exemple d'implantació. CERT. Java RMI és un sistema d'objectes distribuïts ja estudiat en segon curs. També se cita en TSR.
E	Totes les anteriors.
F	Cap de les anteriors.

13. Les cinc dimensions de l'escalabilitat (incloent les que estenen a les bàsiques) són...

A	Temporal, geogràfica, organitzacional, de seguretat i horitzontal. FALS. No hi ha escalabilitat temporal ni escalabilitat de seguretat.
B	Grandària, volum, interfície, xarxa i administrativa. FALS. No hi ha escalabilitat de volum, d'interfície ni de xarxa.
C	Grandària, distància, administrativa, horitzontal i vertical. CERT. Les dimensions bàsiques són l'escalabilitat de grandària, de distància i administrativa. Aquestes tres ja van ser descrites en CSD. L'escalabilitat horitzontal i la vertical són dues dimensions addicionals que estenen l'escalabilitat de grandària.
D	Personal, energètica, administrativa, vertical i memòria. FALS. No hi ha escalabilitat personal, escalabilitat energètica ni escalabilitat de memòria.
E	Totes les anteriors.
F	Cap de les anteriors.

14. Considere un problema en el qual cada instància consta d'una col·lecció de registres. Cada registre pot ser processat amb independència dels altres. Quin tipus d'escalabilitat preferiria per a adaptar-se a instàncies arbitràriament grans d'aquest problema?

A	Escalabilitat vertical. FALS. L'escalabilitat vertical consisteix a reemplaçar components d'un únic ordinador per a augmentar la seua capacitat de còmput. Si les instàncies del problema poden ser arbitràriament grans, arribarà un moment en què l'escalabilitat vertical serà insuficient per a resoldre-ho.
B	Escalabilitat horitzontal. VERTADER. L'escalabilitat horitzontal consisteix a afegir nous nodes al sistema, millorant així el seu rendiment. Si cada registre pot processar-se amb independència de la resta, bastarà amb dividir tota la col·lecció de registres entre els nodes disponibles. Quants més nodes hi haja, menor treball haurà de fer cadascun d'ells. Per tant, l'escalabilitat horitzontal és idònia per a resoldre aquest tipus de problema.
C	Escalabilitat obliqua. FALS. No existeix aquest tipus d'escalabilitat.
D	Escalabilitat diagonal. FALS. No existeix aquest tipus d'escalabilitat.
E	Totes les anteriors.
F	Cap de les anteriors.

15. El conjunt de tècniques que permet aconseguir escalabilitat de grandària inclou...

A	Ús de "*caches*". CERT. Quan s'utilitza una memòria local en els processos clients no resulta necessari contactar amb els servidors en cas que les operacions sol·licitades solament impliquen una lectura de certa part de l'estat gestionat pel servidor. D'aquesta manera els servidors poden atendre a un major nombre de clients i millora l'escalabilitat de grandària.
B	Dades estructurades (SQL). FALS. Que les dades s'emmagatzemen o no de manera estructurada no té cap influència directa sobre l'escalabilitat de grandària. Normalment l'ús de SQL com a llenguatge d'interrogació suposa que la base de dades subjacent siga relacional (i, per tant, estructurada) i que s'utilitzen transaccions ACID per a accedir a la informació. Aquest tipus de transaccions utilitza un control de concurrència implícit per a garantir aïllament serialitzable. Aquest aïllament sol conduir a bloquejos perllongats en cas que múltiples transaccions conflictives tracten d'accedir a un mateix registre. Els bloquejos redueixen tant el rendiment com l'escalabilitat.
C	Semàfors. FALS. L'ús de mecanismes de control de concurrència suposa bloquejos de les activitats concurrents. Aquests bloquejos redueixen l'escalabilitat.
D	Magatzems de lectura-escriptura. FALS. Aquest terme és excessivament ampli. Hi ha molts tipus de magatzems d'informació que permeten accessos de lectura i escriptura. Per si mateix un magatzem de dades no influeix sobre l'escalabilitat. Alguns dels mecanismes que s'utilitzen en el magatzem podran tenir efectes positius (per exemple, el "*sharding*") o negatius (per exemple, els mecanismes de control de concurrència) sobre l'escalabilitat però el magatzem, de manera tan general, no en té cap.
E	Totes les anteriors.
F	Cap de les anteriors.

16. Sobre els sistemes elàstics...

A	No tenen per què ser escalables. FALS. Un sistema distribuït es diu que és elàstic quan és escalable i adaptable. Per tant, forçosament serà escalable.
B	Necessiten ser monitoritzats. CERT. Perquè siguen adaptables han de ser monitoritzats. El monitoratge avalua la càrrega actual suportada pel sistema. Si la càrrega és excessivament lleugera, es podran eliminar instàncies servidores. Per contra, quan la càrrega supere cert llindar superior interessarà afegir rèpliques servidores (escalabilitat horitzontal) perquè el nivell de càrrega en cada node servidor baixe i no hi haja perill de saturar el servei.
C	És raonable gestionar manualment el seu escalabilitat. FALS. Per a qualificar a un sistema distribuït com a adaptable, el seu escalabilitat ha de gestionar-se automàticament.
D	Només poden ser implantats sobre infraestructures *cloud*. FALS. No és estrictament necessari el desplegament sobre un sistema de computació en el núvol per a obtenir un servei elàstic.
E	Totes les anteriors.
F	Cap de les anteriors.

17. …són exemples de models de fallada.

A	Caiguda, omissió d'enviaments, bizantins... CERT. Els models de fallada enumerats en classe van ser: parada, caiguda, omissió d'enviaments, omissió de recepcions, omissió general i bizantí (o arbitrari).
B	Memòria, autoritat... FALS. Veure justificació del primer apartat.
C	Recol·lecció d'informació, seguretat, disponibilitat... FALS. Veure justificació del primer apartat.
D	Col·lisió, corrupció, pèrdua... FALS. Veure justificació del primer apartat.
E	Totes les anteriors.
F	Cap de les anteriors.

18. La robustesa (“dependability”)...

A	Té una definició precisa que genera una magnitud en el rang 0..1. FALS. La robustesa consta de diversos aspectes: fiabilitat, disponibilitat, mantenibilitat, seguretat... Alguns d'ells són quantitatius i poden expressar-se com una probabilitat o valor numèric en el rang 0..1. No obstant això, hi ha uns altres (“security”) que són qualitatius i no poden expressar-se numèricament. Encara que alguns aspectes siguen quantitatius no existeix cap fórmula que permeta combinar els seus valors per a construir cert valor global amb el qual expressar el grau de robustesa d'un servei o sistema.
B	Inclou fiabilitat amb un pes del 50%. FALS. Cap dels aspectes quantitatius de la robustesa té un pes amb el qual participe en el valor global de robustesa d'un sistema.
C	Inclou disponibilitat amb un pes del 35%. FALS. Cap dels aspectes quantitatius de la robustesa té un pes amb el qual participe en el valor global de robustesa d'un sistema.
D	Inclou seguretat. CERT. Un dels aspectes de la robustesa és la seguretat.
E	Totes les anteriors.
F	Cap de les anteriors.

19. Seleccione aquella alternativa amb el major conjunt de models de consistència compatibles amb l'execució següent. Aquest conjunt ha de contenir únicament models compatibles amb l'execució:
P1:W(x)1, P2:W(x)2, P3:R(x)1, P3:W(x)3, P2:W(x)4, P4:R(x)2, P4:R(x)3, P4:R(x)4, P5:R(x)2, P4:R(x)1, P5:R(x)3, P5:R(x)4, P5:R(x)1, P3:R(x)3

A	Seqüencial, "*cache*", processador, causal, FIFO. FALS. No compleix el model causal. Tampoc pot ser seqüencial perquè no és causal. El model causal exigeix que es mantinguen les dependències entre escriptures i lectures, i entre accessos realitzats en un mateix procés. En aquest exemple, es donarien les següents dependències causals: - El valor 1 ha de ser accedit en tots els processos abans que el valor 3, ja que P3 va llegir 1 abans d'escriure 3. Dependència 1 -> 3. - El valor 2 ha de ser accedit en tots els processos abans que el valor 4, ja que P2 va escriure 2 abans d'escriure 4. Dependència 2 -> 4. No obstant això, tant P4 com a P5 observen la seqüència 2, 3, 4, 1 que viola la primera dependència que hem citat. Aquesta execució tampoc arriba a ser "*cache*", ja que en el model "*cache*" s'exigeix que tots els lectors observen la mateixa seqüència de valors llegits. En aquesta execució hi ha tres processos lectors, que obtenen les següents seqüències: P4: 2, 3, 4, 1. P5: 2, 3, 4, 1. P3: 1, 3. L'ordre en què P3 ha llegit els valors 1 i 3 és diferent al que han observat P4 i P5. Per tant, l'execució no és "*cache*". Si no és "*cache*", tampoc podrà ser processador. Per a complir el model processador han de satisfer-se simultàniament totes les restriccions dels models "*cache*" i FIFO. Finalment, aquesta execució sí és FIFO. Perquè ho siga n'hi ha prou que els lectors observen seqüències de valors llegits consistents amb l'ordre en què cada procés haja realitzat les seues pròpies escriptures. Només hi ha un procés que escriga més d'un valor. És P2 i va escriure els valors 2 i 4, en aquest ordre. Els lectors que observen els dos valors han d'obtenir-los en el mateix ordre. Això ocorre tant en P4 com en P5. P3 també és lector però no ha arribat a veure cap dels dos valors i assumirem que mai arribarà a veure'ls. Per tant, l'execució és FIFO.
B	"*Cache*". FALS. Veure justificació del primer apartat.
C	Causal, FIFO. FALS. Veure justificació del primer apartat.
D	Processador, FIFO. FALS. Veure justificació del primer apartat.
E	Processador, "*cache*", FIFO. FALS. Veure justificació del primer apartat.
F	FIFO. CERT. Veure justificació del primer apartat.

20. Seleccione aquella alternativa amb el major conjunt de models de consistència compatibles amb l'execució següent. Aquest conjunt ha de contenir únicament models compatibles amb l'execució:
P1:W(x)1, P2:W(x)2, P3:R(x)1, P3:W(x)3, P2:W(x)4, P4:R(x)1, P4:R(x)3, P4:R(x)4, P5:R(x)1, P4:R(x)2, P5:R(x)3, P5:R(x)4, P5:R(x)2, P2:R(x)1, P3:R(x)3

A	FIFO. FALS. P2 arriba a escriure els valors 2 i 4, en aquest ordre. Perquè es complira el model FIFO tots els processos lectors haurien d'observar els dos valors en el mateix ordre. No obstant això, P4 observa la seqüència 1, 3, 4, 2 on s'ha invertit aquest ordre d'escriptura. Per tant, la consistència no és FIFO.
B	"Cache". CERT. Perquè es complisca la consistència "*cache*" tots els lectors han d'observar la mateixa seqüència de valors llegits. Existeixen quatre processos que arriben a llegir la variable x. Les seqüències observades per cadascun d'ells són: P4: 1, 3, 4, 2. P5: 1, 3, 4, 2. P2: 1. P3: 1, 3. Com pot observar-se tant P4 com P5 observen la mateixa seqüència. P2 i P3 presenten prefixos de la seqüència i, per tant, no la "trenquen". Així, l'execució és "*cache*".
C	Causal, FIFO. FALS. Si una execució no és FIFO, mai podrà ser causal.
D	Causal, "*cache*". FALS. Ja hem vist en l'apartat anterior que no és causal.
E	Processador, causal. FALS. No és causal. A més, per no ser FIFO tampoc podrà ser "processador".
F	Processador, causal, "*cache*", FIFO. FALS. Només és "*cache*".

Segon parcial

1. Un acoblament alt entre dos mòduls A i B...

A	Indica que els mòduls A i B són independents entre si. FALS. Els mòduls serien independendientes en cas que l'acoblament fóra baix: acoblament per dades.
B	Facilita que A es recupere de les fallades de B. FALS. L'acoblament mesura les dependències existents entre dos mòduls. Com més alt siga l'acoblament majors dependències hi haurà entre els mòduls. Les dependències complicarien la recuperació.
C	Permet que A interactue amb B només quan siga estrictament necessari. FALS. De nou, això ocorreria en tenir un acoblament el més baix possible (nivell d'acoblament per dades).
D	Minimitza l'ús de la comunicació de xarxa entre A i B. FALS. De nou, això ocorreria en tenir un acoblament el més baix possible (nivell d'acoblament per dades).
E	Totes les anteriors.
F	Cap de les anteriors.

2. … són nivells d'acoblament.

A	Contingut, control... CERT. Els nivells d'acoblament enumerats en classe són: (1) contingut, (2) control, (3) comú, (4) per estructures de dades i (5) per dades.
B	Fort, feble... FALS. Aquests no són nivells d'acoblament sinó adjectius que serveixen per a qualificar de manera general l'acoblament.
C	Alt, baix... FALS. Aquests no són nivells d'acoblament sinó adjectius que serveixen per a qualificar de manera general l'acoblament.
D	Xarxa, memòria... FALS. No són nivells d'acoblament.
E	Totes les anteriors.
F	Cap de les anteriors.

3. Sobre la persistència en una aplicació distribuïda...

A	Resulta necessària si l'estat ha de superar certes situacions de fallada. CERT. Per exemple, si se segueix un model de fallades de caiguda i els components poden arribar a recuperar-se interessarà guardar certa part de l'estat en emmagatzematge persistent o en altres rèpliques del component. Així la informació persistida superarà aquestes fallades de caiguda i permetrà que el component es recupere més ràpid.
B	Necessita emmagatzematge secundari per a implantar-se de manera adequada. FALS. Si els components es repliquen, el seu estat persistirà a pesar que alguns components fallen. Per tant, pot arribar a implantar-se la persistència sense necessitat de recórrer al emmagatzematge secundari.
C	L'emmagatzematge persistent proporciona consistència seqüencial. FALS. La consistència depèn dels protocols utilitzats per a propagar les modificacions d'estat a totes les rèpliques de l'element modificat. No depèn de l'emmagatzematge persistent utilitzat sinó de com es propaguen les escriptures i quan es permeta que un lector consulte l'estat.
D	Els magatzems no persistents no poden superar cap situació de fallada. FALS. Es poden superar múltiples situacions de fallada replicant els components, fins i tot utilitzant magatzems no persistents per a l'estat gestionat.
E	Totes les anteriors.
F	Cap de les anteriors.

4. Sobre els magatzems clau-valor...

A	Són un exemple de magatzem de dades NoSQL. CERT. Existeixen tres tipus principals de magatzems NoSQL: clau-valor, de documents i de registres extensibles. Per tant, els magatzems clau-valor són tant un tipus com un exemple de magatzems de dades NoSQL.
B	El seu esquema està format per tres atributs: clau, valor i atomicitat. FALS. El seu esquema només consta de dos camps o atributs: la clau i el valor. L'atomicitat és una garantia de les transaccions en el model relacional. No és un atribut d'un esquema.
C	VoltDB és un exemple d'ells. FALS. VoltDB és un exemple de base de dades relacional. Per tant, utilitza SQL com a llenguatge de consulta i no és un magatzem clau-valor (ni tan sols és un magatzem NoSQL).
D	Permeten consultes utilitzant atributs no primaris. FALS. Les consultes únicament poden realitzar-se a través de l'atribut clau i aquest es considera "primari".
E	Totes las anteriors.
F	Cap de les anteriors.

5. Sobre MongoDB…

A	És un exemple de magatzem NoSQL. CERT. És un exemple de magatzem de documents. Els magatzems de documents són un tipus de magatzem NoSQL.
B	Utilitza "*sharding*" per a millorar la seua disponibilitat. FALS. Utilitza "*sharding*" per a millorar l'escalabilitat.
C	Utilitza només replicació per a millorar la seua escalabilitat. FALS. Utilitza "*sharding*" per a millorar l'escalabilitat.
D	És un exemple de magatzem clau-valor. FALS. No és un magatzem clau-valor, sinó un magatzem de documents escalable. Els magatzems de documents i els magatzems clau-valor presenten característiques diferents.
E	Totes les anteriors.
F	Cap de les anteriors.

6. La seguretat...

A	És una propietat quantitativa dels sistemes distribuïts. FALS. En aquest segon parcial la seguretat es va explicar en el tema 7 com un aspecte qualitatiu dels sistemes distribuïts robusts.
B	És un aspecte de la fiabilitat. FALS. És un aspecte de la robustesa ("dependability").
C	Sempre necessita l'ús de tècniques de xifrat. FALS. El xifrat és un dels mecanismes possibles (a utilitzar per a garantir confidencialitat, per exemple) però se'n poden utilitzar uns altres.
D	Té com a objectiu la distinció entre usuaris normals i administradors del sistema. FALS. Els objectius principals de la seguretat són: confidencialitat, integritat, disponibilitat i comptabilitat. En cap d'ells s'intenta només distingir entre aquests dos rols.
E	Totes les anteriors.
F	Cap de les anteriors.

7. ...són exemples d'estratègies defensives.

A	L'aïllament i l'exclusió... CERT. L'aïllament és un exemple d'estratègia defensiva d'àmbit ampli. Per la seua banda, l'exclusió és un exemple d'estratègia defensiva d'àmbit mitjà. També es van presentar la restricció, la recuperació i la penalització com a estratègies defensives específiques.
B	El xifrat i les llistes de control d'accés... FALS. Tant el xifrat com les llistes de control d'accés són mecanismes de seguretat. No són estratègies defensives.
C	Les contrasenyes i els "*nonces*"... FALS. Tant les contrasenyes com els "*nonces*" són mecanismes de seguretat. No són estratègies defensives.
D	Les capacitats... FALS. Les capacitats són mecanismes de seguretat. No són estratègies defensives.
E	Totes les anteriors.
F	Cap de les anteriors.

8. Els sistemes distribuïts...

A	Faciliten l'especificació d'una TCB. FALS. La TCB ha de ser el més simple possible. És més senzill definir-la per a un sistema format per un únic ordinador que per a un sistema distribuït on participen múltiples nodes.
B	Impossibiliten els atacs "man-in-the-middle". FALS. Al contrari, és un atac d'accés que requereix redirecció o intercepció del tràfic de paquets que circulen per la xarxa. Si no hi haguera comunicació a través de la xarxa (com ocorre en un sistema distribuït) seria impossible realitzar un atac d'aquest tipus.
C	Dificulten la garantia de discreció / secret. CERT. Un sistema distribuït està exposat a transmetre informació rellevant a través de la xarxa de comunicacions. En aquests entorns resulta difícil ocultar aquesta transmissió d'informació.
D	Eviten la falsificació / corrupció de les dades. FALS. En un sistema distribuït s'ha de transmetre informació entre els diferents agents. Sempre existirà risc de falsificació o corrupció de la informació transmesa.
E	Totes les anteriors.
F	Cap de les anteriors.

9. Siguen A i B dos exemples de "text pla". Siga *h* una funció hash criptogràfica...

A	h(A) = h(B) si i només si A = B. FALS. Les funcions hash criptogràfiques han de ser unidireccionals, però no necessiten ser bijectives. Per tant, quan A siga diferent a B, serà improbable que h(A) = h(B), però no es podrà assegurar que h(A) siga diferent d'h(B).
B	Donat h(A), és computacionalment senzill obtenir A. FALS. Al contrari, donat h(A) ha de ser computacionalment difícil i costós obtenir A.
C	*h* és una funció bijectiva. FALS. És quasi equivalent al que s'enuncia en el primer apartat.
D	*h* pot utilitzar-se per a implantar MACs. CERT. Les funcions hash criptogràfiques poden ser utilitzades per a implantar codis d'autenticació de missatges (MAC).
E	Totes les anteriors.
F	Cap de les anteriors.

10. Els protocols de xifrat (criptogràfics)...

A	Eviten els atacs "man-in-the-middle". FALS. No sempre aconseguiran evitar-los. Qui realitze aquests atacs pot haver trobat la manera de desxifrar els missatges transmesos (esbrinant la clau necessària per a realitzar el desxifrat).
B	Impossibiliten la reexpedició de missatges com a tècnica d'atac. FALS. Dependrà de quin fóra l'objectiu del missatge que s'estiga reexpedint. A més, l'atacant (igual que en el cas anterior) podria haver esbrinat quina era la clau necessària per a desxifrar.
C	Necessiten xifrat asimètric. FALS. El xifrat simètric també pot utilitzar-se en els protocols criptogràfics.
D	Les seues propietats de seguretat depenen de l'esquema de distribució de claus utilitzat. CERT. Cal evitar que usuaris desautoritzats coneguen les claus. Això depèn de l'esquema de distribució de claus.
E	Totes les anteriors.
F	Cap de les anteriors.

11. Sobre els programes i serveis...

A	Tots els programes són serveis. FALS. Mentre un programa o aplicació no haja sigut desplegat i es mantinga en execució no serà un servei.
B	El desplegament de programes genera serveis. CERT. Un servei és un programa desplegat i actiu.
C	Els serveis només poden ser facilitats per programes. FALS. No tots els serveis són responsabilitat exclusiva dels programes.
D	Tots els serveis són també programes. FALS. Per desgràcia no tot està automatitzat i hi haurà alguns serveis que encara hauran de ser prestats per operadors o administradors dels sistemes utilitzats.
E	Totes les anteriors.
F	Cap de les anteriors.

12. Sobre el SLA d'un servei...

A	Consta principalment d'aspectes de rendiment, disponibilitat i funcionalitat. CERT. Aquests són els seus aspectes principals.
B	Condiciona la ubicació dels components d'un servei durant el desplegament. FALS. El rendiment, disponibilitat i funcionalitat d'un servei no depenen de la ubicació dels seus components.
C	Determina la tecnologia de programació utilitzada per a implantar els components del servei. FALS. Els desenvolupadors d'un servei tenen llibertat per a decidir les tecnologies a utilitzar. Això no es veu afectat pel SLA. De fet, el SLA sol establir-se una vegada el servei ja està desenvolupat i no condiciona aquestes decisions prèvies.
D	Determina l'IaaS utilitzat per a desplegar el servei. FALS. El SLA d'un servei no sempre determinarà quin IaaS haurà d'utilitzar-se ni les característiques que aquest hauria de complir. No sempre és la infraestructura el que s'ofereix com a servei a l'hora d'establir un SLA.
E	Totes les anteriors.
F	Cap de les anteriors.

13. Exemples d'elements de la gestió del cicle de vida d'un servei:

A	El desplegament inicial.
B	L'actualització de components.
C	Els ajustos per a millorar l'escalabilitat.
D	Les reconfiguracions.
E	Totes les anteriors. Tot el que s'esmenta en els apartats anteriors són fases del cicle de vida que han de gestionar-se adequadament en un servei.
F	Cap de les anteriors.

14. ...és una part d'un descriptor de desplegament.

A	La política de seguretat... FALS. Les polítiques de seguretat no solen estar incloses en els descriptors de desplegament.
B	La resolució de dependències... CERT. El descriptor de desplegament incloïa els següents elements: (1) les plantilles de configuració emplenades per a cadascuna de les instàncies (de cada component), (2) el pla de desplegament emplenat, en el qual s'especifica en quin node se situarà cada instància, (3) la resolució o enllaç de dependències, tant internes com a externes. Per tant, aquest últim element és el que fa certa aquesta afirmació.
C	El codi dels components... FALS. El codi dels components no ha d'incloure's en un descriptor de desplegament.
D	La descripció de l'API implantada... FALS. La descripció d'una API pot ser necessària a l'hora de desenvolupar altres components que interactuen amb aquell que facilite l'API. El desplegament es duu a terme una vegada els components ja estan implantats. No té sentit incloure les descripcions de les API en un descriptor de desplegament.
E	Totes les anteriors.
F	Cap de les anteriors.

15. La injecció de dependències...

A	S'implanta regularment en els entorns de contenidors.
B	Desacobla el codi de l'aplicació de la implantació concreta de les dependències.
C	Crea un graf d'instàncies de components.
D	És una tècnica que evita preocupar-se per les dependències del codi.
E	Totes les anteriors. CERT. Totes les afirmacions anteriors es compleixen quan analitzem la injecció de dependències entre instàncies de components durant el desplegament d'un servei.
F	Cap de les anteriors.

16. La semàntica "almenys una vegada"...

A	Garanteix que una petició s'execute exactament una vegada en el servidor. FALS. La semàntica "almenys una vegada" garanteix que les operacions arriben com a mínim una vegada al servidor. Pot haver-hi repeticions. Si es pretén garantir una semàntica "exactament una vegada" caldrà complementar els mecanismes que ja proporcionaven la semàntica "almenys una vegada" amb altres mecanismes que detecten les repeticions en els missatges de petició i retornen el resultat original a aquestes peticions, sense necessitat de reexecutar l'operació. Això comporta que s'emmagatzemen els resultats de les operacions ja executades i que s'identifiquen correctament les peticions, permetent la detecció de les repeticions. Tant l'emmagatzematge com la identificació no resulten necessaris en una semàntica "almenys una vegada".
B	Evita que una petició genere canvis d'estat en el servidor. FALS. L'objectiu d'aquesta semàntica no és evitar que es generen canvis d'estat. El seu objectiu és fer arribar als servidors totes les peticions iniciades pels clients, evitant que alguna d'elles es perda. Si les operacions associades a les peticions impliquen un canvi d'estat en el servidor, aquest canvi d'estat es farà sense cap problema.
C	S'utilitza en el patró PUSH-PULL. FALS. El patró PUSH-PULL no garanteix un lliurament fiable dels missatges enviats. És un patró unidireccional i asincrònic que pot patir pèrdues de missatges. Per tant, no garanteix la semàntica "almenys una vegada".
D	S'utilitza en el patró PUB-SUB. FALS. El patró PUB-SUB no garanteix un lliurament fiable dels missatges enviats. És un patró de difusió unidireccional i asincrònic que pot patir pèrdues de missatges. Per tant, tampoc garanteix la semàntica "almenys una vegada".
E	Totes les anteriors.
F	Cap de les anteriors.

17. El patró PUB-SUB bàsic...

A	És un patró de multienviament (difusió).
B	La informació flueix des del publicador als subscriptors.
C	No garanteix el lliurament dels missatges.
D	Permet que els subscriptors filtren els missatges que reben.
E	Totes les anteriors. CERT. Les quatre afirmacions anteriors proporcionen les principals característiques d'aquest patró arquitectònic.
F	Cap de les anteriors.

18. El patró PUB-SUB bàsic...

A	És intrínsecament escalable. FALS. No sempre podrà escalar amb facilitat. El seu objectiu és difondre missatges a cert conjunt de processos subscriptors. Es podrà aconseguir un bon rendiment i escalabilitat si la xarxa subjacent permet realitzar difusions fàcilment (per exemple, quan la xarxa física admeta adreces de difusió) però en altres casos podrà no ser escalable.
B	Reenvia missatges antics als nous subscriptors. FALS. Precisament un dels inconvenients d'aquest patró és que els nous subscriptors no arriben a rebre els missatges difosos abans de la seua subscripció (missatges "antics").
C	Exigeix un conjunt de subscriptors estàtic. FALS. Com suggereix l'apartat anterior, els processos poden subscriure's quan ho consideren convenient. Pot haver-hi "nous subscriptors". Per tant, el conjunt de subscriptors és dinàmic.
D	Sempre exigeix que els subscriptors es configuren amb la URL del publicador. FALS. És una recomanació i és la forma normal de configurar-ho ja que el publicador sol ser l'agent estàtic en aquest patró. No obstant això, no s'exigeix aquest tipus de configuració.
E	Totes les anteriors.
F	Cap de les anteriors.

19. Quan múltiples clients es connecten a múltiples servidors...

A	Es garanteix la semàntica “almenys una vegada” per a les peticions. FALS. No es garanteix aquesta semàntica. La semàntica utilitzada dependrà de com gestionen tant el client com el servidor la reexpedició o múltiple recepció de peticions en cas que hi haja hagut errors en la transmissió.
B	Es garanteix la semàntica “com a màxim una vegada” per a les peticions. FALS. No es garanteix aquesta semàntica. La semàntica utilitzada dependrà de com gestionen tant el client com el servidor la reexpedició o múltiple recepció de peticions en cas que hi haja hagut errors en la transmissió.
C	Les peticions idempotents s'executen només una vegada. FALS. Les peticions idempotents poden executar-se tantes vegades com es vulga, ja que sempre generen el mateix resultat. Per tant, no té sentit que s'exigisca executar-les una sola vegada.
D	Quan s'utilitze una cua intermèdia, els clients i servidors poden configurar-se amb la URL d'aquesta cua, simplificant la seua configuració. CERT. La cua intermèdia sol ser l'element estable en aquest patró. Per tant, és convenient que la resta d'agents es configuren amb la seua URL.
E	Totes les anteriors.
F	Cap de les anteriors.

20. Els patrons PUSH-PULL i PUB-SUB…

A	Són (tots dos) patrons de comunicació unidireccionals. CERT. En el PUSH-PULL els missatges s'envien des del socket PUSH i arriben al socket PULL. En el patró PUB-SUB els missatges s'envien des del socket PUB i arriben al socket SUB. Tots dos són unidireccionals.
B	Són (tots dos) patrons de difusió. FALS. El patró PUSH-PULL no empra difusions. El patró PUB-SUB sí que és un patró de difusió. El socket PUB difon els missatges cap als sockets SUB.
C	Suspenen a l'emissor fins que els receptors obtinguen els missatges enviats. FALS. Són patrons de comunicació asincrònica. No suspenen a l'emissor en cap cas.
D	Garanteixen que no hi haja pèrdues de missatges. FALS. Tots dos són patrons de comunicació no fiable en els quals poden perdre's missatges.
E	Totes les anteriors.
F	Cap de les anteriors.