Este examen tiene una duración total de 2 horas y 30 minutos. Consta de dos partes: TEORIA y PRÁCTICAS. Debe escribir su nombre y apellidos en la hoja de respuestas **de cada parte**.

PARTE TEORIA

Esta parte tiene una puntuación máxima de **10 puntos**, que equivalen a **3** puntos de la nota final de la asignatura. Indique, para cada una de las siguientes **50 afirmaciones**, si éstas son verdaderas (**V**) o falsas (**F**).

Cada respuesta vale: correcta= 0.2, errónea= -0.2, vacía=0.

Importante: Los **primeros 3 errores no penalizarán**, de modo que tendrán una valoración equivalente a la de una respuesta vacía. A partir del 4º error (inclusive), sí se aplicará el decremento por respuesta errónea.

Respecto a la sincronización en sistemas distribuidos: V/F

1.	Si los nodos de un sistema distribuido están dispuestos en un anillo lógico, resultaría sencillo obtener el estado global del sistema, pues pueden registrar su estado cuando reciben el token que circula por el anillo. *JUSTIFICACIÓN: El estado global incluye no sólo el estado de cada nodo, sino también los mensajes en tránsito (que han sido enviados y que todavía no han llegado a su destino). Por tanto, haría falta registrar también (de algún modo) cuáles son dichos mensajes, de modo que obtengamos una instantánea consistente.*	F
2.	En un sistema distribuido, si varios nodos desean hacer uso de un recurso compartido, pero dicho recurso no puede ser utilizado a la vez por más de un nodo, debemos utilizar un algoritmo de selección de líder para escoger al nodo que puede hacer uso del recurso. *JUSTIFICACIÓN: Para el acceso en exclusión mutua a un recurso compartido debemos utilizar un algoritmo de exclusión mutua de sistemas distribuidos (por ejemplo, cualquiera de los tres algoritmos vistos en la asignatura, i.e. algoritmo centralizado, algoritmo distribuido o algoritmo para anillos).*	F
3.	El algoritmo de Chandy-Lamport permite establecer un orden total entre los eventos de un sistema distribuido. *JUSTIFICACIÓN: El algoritmo Chandy-Lamport permite obtener el estado global del sistema distribuido. Para establecer un orden total, utilizaremos los relojes lógicos de Lamport y los identificadores de los nodos.*	F

Respecto a los servicios Web RESTful:

4.	Son atendidos por servidores que mantienen el estado de las peticiones de los clientes. *JUSTIFICACIÓN: Los servicios son "sin estado", de modo que los servicios no mantienen ninguna sesión respecto a las peticiones de los clientes.*	F
5.	`GET https:/weatherapp.com/zipcodes` es un ejemplo de llamada a un servicio Web RESTFul.	V
6.	En un servicio Web RESTFul, los mensajes constan de una cabecera formada por varios campos fijos definidos por el estándar RESTful, un conjunto de propiedades que pueden ser definidas por la aplicación y de un contenido (generalmente, en XML o JSON). *JUSTIFICACIÓN: El estilo arquitectónico REST no especifica ninguna cabecera, ni campos fijos definidos. Por tanto, los mensajes tienen estructura libre.*	F
7.	Un sistema distribuido con comunicación basada en servicios web REST emplea generalmente comunicación sincrónica no persistente y direccionamiento directo.	V

Respecto al mecanismo de comunicación Java Message Service:

8.	Existen dos tipos de destinos: Colas (Queues), utilizadas para enviar mensajes a un único cliente; y Temas (Topics), que permiten la publicitación/suscripción.	V
9.	Las factorías de conexiones y los destinos se crean utilizando una herramienta administrativa ofrecida por el Proveedor de JMS.	V
10.	JMS utiliza direccionamiento directo, pues en la cabecera del mensaje se especifica claramente el destino del mensaje. *JUSTIFICACIÓN: JMS emplea direccionamiento indirecto, ya que el cliente envía el mensaje a un destino (Destination), que puede ser una cola (Queue) o un tema (Topic). Y no tiene por qué conocer quién será el consumidor del mensaje.*	F
11.	Java Message Service ofrece una comunicación fuertemente acoplada, pues tanto el productor como el consumidor de un mensaje se crean utilizando la misma factoría de conexión. *JUSTIFICACIÓN: JMS ofrece una comunicación débilmente acoplada, ya que al emplear destinos, el productor y el consumidor del mensaje no necesitan conocerse. Solamente requieren conocer el destino (y estar de acuerdo en el formato del mensaje).*	F

Sobre el mecanismo de comunicación ROI:

12.	Cuando se pasa un objeto por valor, se copia una referencia al mismo. JUSTIFICACIÓN: *Cuando se pasa un objeto por valor, el estado del objeto se empaqueta, mediante un proceso denominado serialización.*	F
13.	El proxy empaqueta los argumentos una vez recibe la contestación del esqueleto. JUSTIFICACIÓN: *El proxy desempaqueta los argumentos al recibir la contestación, para así devolverlos al proceso cliente.*	F
14.	El componente denominado ORB se encarga, entre otras cosas, de identificar y localizar a los objetos remotos.	V
15.	El middleware de un sistema con comunicación basada en ROI emplea generalmente comunicación sincrónica no persistente y direccionamiento directo.	V

Sobre el mecanismo de comunicación Java RMI:

16.	No proporciona transparencia de ubicación, porque la sintaxis de invocación del método es distinta dependiendo de si el objeto es local o remoto. *JUSTIFICACIÓN: La sintaxis de invocación es la misma. En la interfaz del objeto definimos los métodos del objeto (con independencia de que vaya a ser remoto o local). Si el objeto es remoto, la interfaz debe extender java.rmi.Remote.*	F
17.	Se considera objeto local todo objeto que sólo puede invocarse desde la computadora en que se define (aunque sea desde otras máquinas virtuales Java), y objeto remoto a todo objeto que puede invocarse desde otras computadoras. *JUSTIFICACIÓN: Si un objeto se invoca desde otras máquinas virtuales Java de la misma computadora, también se considera como objeto remoto.*	F
18.	El proxy correspondiente a un objeto remoto se crea en tiempo de ejecución cuando se accede por primera vez al objeto remoto.	V
19.	La clase de un objeto remoto debe implementar un interfaz que extienda *java.rmi.Remote*.	V

Sobre los sistemas distribuidos:

20.	Para mejorar la escalabilidad de tamaño, los clientes deben delegar en el servidor tantas responsabilidades como sea posible. JUSTIFICACIÓN: *Al contrario, contra menos se centralicen las tareas en los servidores, mayor escalabilidad de tamaño se podrá conseguir.*	F
21.	Los algoritmos descentralizados facilitan distribuir la carga computacional entre diferentes ordenadores.	V

22.	La transparencia de acceso oculta las diferencias en la representación de los datos y en cómo se accede a los recursos.	V
23.	La *capa de middleware*, que se ubica bajo el nivel de aplicación, puede integrar algunos mecanismos de comunicación que faciliten la programación de aplicaciones distribuidas; por ejemplo: JMS.	V
24.	Cuando un sistema distribuido es abierto facilita que uno de sus módulos o componentes pueda utilizarse en otro sistema distribuido.	V
25.	Para conseguir escalabilidad de distancia debe asumirse que se está utilizando una red de área local. JUSTIFICACIÓN: *La escalabilidad de distancia permite extender el sistema por redes de área amplia (WAN), por lo que si se usan algoritmos basados en redes de área local, se deben considerar los efectos de los retardos en la transmisión de los datos y la menor fiabilidad de las comunicaciones.*	F
26.	Se requiere utilizar algoritmos descentralizados para conseguir escalabilidad administrativa, de modo que los cómputos se distribuyen entre diferentes áreas administrativas del sistema. JUSTIFICACIÓN: *Para conseguir escalabilidad administrativa se deben utilizar protocolos y mecanismos estándar de autenticación y autorización; así como implmentar mecanismos para proteger a cada organización del resto y del propio sistema.*	F

Respecto a la arquitectura de los sistemas distribuidos:

27.	Los sistemas replicados y los sistemas peer-to-peer presentan distribución horizontal.	V
28.	Las arquitecturas de capas, las arquitecturas basadas en objetos y las arquitecturas basadas en eventos son ejemplos de estilos arquitectónicos de arquitecturas software.	V
29.	En una arquitectura centralizada se distinguen dos tipos de roles: servidor, responsable de la gestión de un recurso determinado y que define una serie de servicios (sobre dicho recurso); y cliente, que es un componente que utiliza dichos servicios.	V

Sobre los algoritmos vistos en esta asignatura de exclusión mutua y elección de líder para sistemas distribuidos:

30.	En el algoritmo de elección de líder para anillos, el mensaje ELECCIÓN incluye dos campos: uno para registrar el iniciador y otro para informar de cúal era el líder antes de iniciar el algoritmo. *JUSTIFICACIÓN: El mensaje ELECCIÓN incluye dos campos. Uno para registrar el iniciador y el otro para registrar la lista de nodos activos.*	F
31.	En el algoritmo distribuido de exclusión mutua, un nodo entra en la sección crítica cuando recibe un mensaje OK de cada uno de los demás nodos.	V
32.	El algoritmo distribuido de exclusión mutua requiere que se utilicen relojes lógicos de Lamport e identificadores para decidir la precedencia de las solicitudes *JUSTIFICACIÓN: Se requiere establecer un orden total de los eventos para así resolver los empates que se puedan producir cuando dos o más nodos desean acceder "a la vez" a la sección crítica.*	V
33.	En el algoritmo Bully, un nodo sabe que será el nuevo coordinador cuando, tras enviar los mensajes de tipo ELECCIÓN, no recibe ninguna respuesta. *JUSTIFICACIÓN: Como envía ELECCIÓN a los nodos con mayor identificador, si no recibe respuesta, entonces él es el nodo activo con mayor identificador. Por tanto, él es el nuevo líder.*	V
34.	En el algoritmo centralizado de exclusión mutua, cuando un nodo solicita entrar en la sección crítica y ésta está ocupada, el coordinador envía un mensaje al nodo que está en la sección crítica para que este último, cuando termine, se lo notifique al nodo que ha hecho la solicitud. *JUSTIFICACIÓN: El coordinador no envía ningún mensaje al nodo en la sección crítica. Cuando éste termina, notifica directamente al coordinador.*	F

Respecto a los relojes lógicos de Lamport y vectoriales:

35.	Si el valor en **x** del reloj vectorial es igual a V(x)=[9,0,1] y el valor en **y** del reloj vectorial es igual a [6,2,2], entonces podemos afirmar que **x || y .**	V
36.	Si el valor en un evento **x** del reloj lógico de Lamport es igual a C(**x**)=1 y el valor en **y** del reloj lógico de Lamport es igual a C(**y**)=6, podemos afirmar que **x→ y**. *JUSTIFICACIÓN: Dados los valores de los relojes lógicos de Lamport, no se puede afirmar si dos eventos son concurrentes o si uno ocurre antes que el otro.*	F

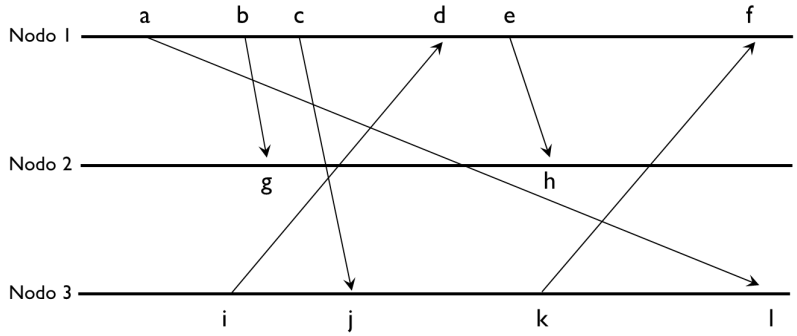

Figura 1: muestra todos los envíos de mensajes entre tres nodos

Respecto a la figura 1:

37.	El valor en **d** del reloj vectorial es igual a V(d)=[4,0,1] y el valor en **j** del reloj vectorial es igual a V(j)= [3,1,2] *JUSTIFICACIÓN: V(d)=[4,0,1] , V(j)=[3,0,2]*	F
38.	Se cumple que **c → l** y que **e || l**	V
39.	El valor en **l** del reloj lógico de Lamport es igual a 6	V

Respecto a los algoritmos de sincronización de relojes físicos.

40.	El algoritmo de Berkeley asume que el envío de un mensaje desde el nodo A al nodo B consume el mismo tiempo que la respuesta desde B hasta A.	V
41.	En el algoritmo de Cristian, si un cliente C pregunta al servidor su hora en el instante 20.000 (según el reloj de C), recibe la respuesta del servidor en el instante 20.010 (según el reloj de C) y calcula que el nuevo valor para su reloj debe ser 20.024, entonces se puede deducir que la respuesta del servidor habrá sido 14. *JUSTIFICACIÓN: En el algoritmo de Cristian, el servidor envía "su valor de reloj", por ejemplo, 20.014. Pero no envía "su diferencia" respecto al cliente.*	F
42.	En el algoritmo de Berkeley, el nodo que actúa como coordinador puede que tenga que ajustar también su reloj, al igual que los otros nodos que participan en el algoritmo.	V

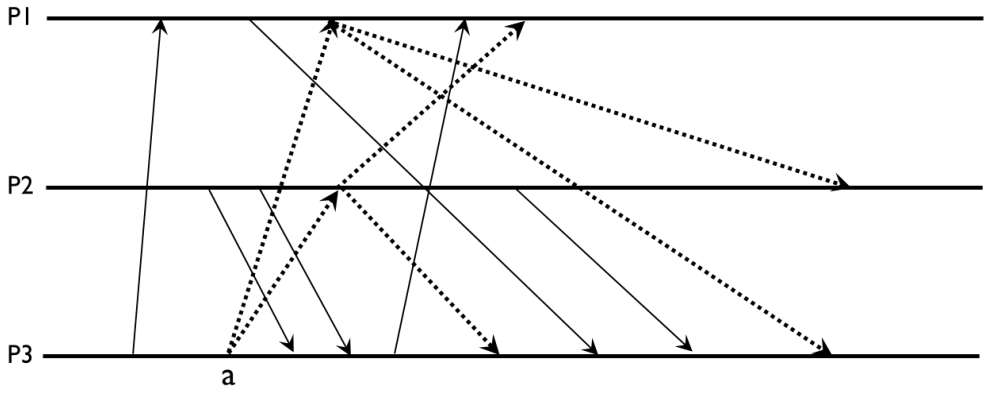

Figura 2

Respecto a la figura 2, suponga que P3 inicia el algoritmo de Chandy-Lamport en **a**, siendo las líneas discontinuas los mensajes que se envían como consecuencia de la ejecución de este algoritmo y las líneas continuas los mensajes normales. Cuando finaliza el algoritmo,

43.	en el canal (P3,P1) se habrán registrado 0 mensajes. *JUSTIFICACIÓN: Al acabar el algoritmo, la instantánea consistente comprende los puntos donde P1 y P2 recibieron el primer MARCA (pues ahí guardaron su estado), así como el punto "a" (donde P3 guardó su estado). Si unimos esos puntos, podemos ver claramente que el algoritmo habrá registrado 1 mensaje en canal (P1,P3), 2 mensajes en canal (P2, P3), 0 mensajes en canal (P3,P1) y 0 mensajes en canal (P3,P2). Los otros mensajes, o bien pasaron antes de la instantánea (como el primer mensaje del dibujo), o bien pasarán después, en el futuro, como ocurre con el mensaje en canal (P3,P1) y el último mensaje del canal (P2, P3).*	V
44.	en el canal (P1,P3) se habrá registrado 1 mensaje.	V
45.	en el canal (P2,P3) se habrán registrado 3 mensajes.	F

Respecto a los sistemas peer-to-peer, los sistemas Grid y los sistemas Cloud:

46.	El *Cloud Computing de Plataforma como Servicio* (PaaS) permite desarrollar, desplegar y ejecutar aplicaciones web a través de Internet.	V
47.	Cuando se utilizan los servicios Cloud, no es necesario sobredimensionar a priori el sistema para prevenir un futuro aumento en la necesidad de recursos en caso de que el sistema necesite crecer. *JUSTIFICACIÓN: Las empresas que proporcionan los servicios Cloud se encargarán de ofrecernos la escalabilidad de tamaño que, como clientes, necesitemos.*	V
48.	En las arquitecturas *peer-to-peer parcialmente centralizadas* el servicio de transferencia de datos se realiza a través de servidores denominados *supernodes* (supernodos). *JUSTIFICACIÓN: Es el servicio de localización de recursos el que se realiza a través de los supernodos.*	F
49.	Aunque los sistemas *peer-to-peer* se basan en distribución horizontal, para el servicio de localización de recursos se han realizado implementaciones con uno o con varios servidores centralizados, a los que dirigir las búsquedas de recursos.	V
50.	Los cuatro objetivos de los sistemas distribuidos se observan claramente en los sistemas Grid: permiten el acceso a recursos remotos compartidos; ofrecen transparencia (por ejemplo, de acceso, de localización); son sistemas abiertos y ofrecen escalabilidad (por ejemplo, de distancia).	V

PARTE PRACTICAS

Esta parte tiene una puntuación máxima de **10 puntos**, que equivalen a **1** punto de la nota final de la asignatura. Indique, para cada una de las siguientes **16 afirmaciones**, si éstas son verdaderas (**V**) o falsas (**F**).

Cada respuesta vale: correcta= 0.625, errónea= -0.625, vacía=0.

Importante: El **primer error no penalizará**, de modo que tendrá una valoración equivalente a la de una respuesta vacía. A partir del 2º error (inclusive), sí se aplicará el decremento por respuesta errónea.

Respecto a la práctica sobre Active Directory:

1.	En esta práctica se pedía crear un único dominio con dos controladores de dominio.	F
2.	Cada dominio solo puede tener un domain controller (DC, controlador de dominio).	F
3.	Los diferentes dominios de una red corporativa se estructuran de forma jerárquica, formando un solo árbol de dominios.	F
4.	Para crear un dominio secundario del dominio raíz, es necesario proporcionar las credenciales del administrador del dominio raíz.	V
5.	Un usuario de un dominio sólo puede iniciar sesión en ordenadores de su mismo dominio.	F
6.	Para que un usuario pueda modificar un fichero de un recurso compartido, hace falta que tenga permisos para ello tanto sobre el fichero como sobre el recurso compartido.	V
7.	Los administradores de dominios secundarios o subdominios pueden, por defecto, abrir sesión en la maquina controlador del dominio principal.	F
8.	Una vez se ha construido el bosque, solo queda un usuario con capacidades administrativas, el administrador del dominio raíz. El resto de administradores locales se desactivan.	F

Respecto a la práctica del Chat distribuido en Java RMI:

9.	Cada uno de los procesos ChatClient y ChatServer abre un puerto, porque es necesario para utilizar el servidor de nombres (rmiregistry).	F
10.	ChatRobot debe utilizar la clase que proporciona la interfaz gráfica (ChatUI) para obtener las indicaciones sobre a qué servidor y canal debe conectarse.	F
11.	Todos los objetos invocables de forma remota deben estar registrados en el servidor de nombres (rmiregistry).	F
12.	Los objetos de la clase ChatMessage son creados siempre por el ChatServer.	F

Dado el siguiente fragmento de código, que corresponde al código original de la práctica de Chat, donde los puntos suspensivos indican código omitido:

```
// Simple ChatUser implementation.
// Notice how this implementation just calls a listener
public class ChatUser extends UnicastRemoteObject implements IChatUser
{
   private String nick;
   private MessageListener lis;

   public ChatUser (...) throws RemoteException {...}
   public String getNick() throws RemoteException {...}
   public void sendMessage (IChatMessage msg) throws RemoteException {
      lis.messageArrived (msg);
   }
}

public class ChatClient implements MessageListener {
   …
   public String [] doConnect (…) throws Exception {
      …
       Registry reg = LocateRegistry.getRegistry (…);
       srv = (IChatServer) reg.lookup (…);
      …
   }
}
```

13.	Los objetos de la clase ChatClient pueden ser accedidos remotamente.	F
14.	En la clase ChatClient, la variable srv es un proxy del servidor de chat (objeto remoto ChatServer).	V
15.	Los objetos de la clase ChatUser pueden ser accedidos remotamente.	V
16.	El método sendMessage del objeto ChatUser se invoca para que el usuario que representa reciba un mensaje de un canal.	V