TSR: Primer parcial

Este examen comprende 10 preguntas de opción múltiple. En cada caso solamente una respuesta es correcta. Debe responderse en una hoja aparte. Las respuestas correctas aportan 1 punto a la calificación de esta prueba. Las incorrectas reducen la calificación 0.33 puntos.

1. ¿Qué afirmación sobre los sistemas LAMP es correcta?

A	Los componentes en un sistema LAMP se despliegan habitualmente sobre nodos con sistema Windows.
B	Un sistema LAMP suele tener una arquitectura de 3 niveles.
C	Ningún componente en un sistema LAMP puede replicarse.
D	Un sistema LAMP completo no puede desplegarse sobre un único nodo.

2. Una diferencia entre la computación en la nube (CN) y los clústeres de alta disponibilidad (AD) es:

A	Los clústeres AD replican los componentes de los servicios, pero los servicios desplegados en CN no usan replicación.
B	Los clústeres AD se utilizan principalmente para aplicaciones de ciencia de datos; la CN se suele utilizar para desplegar servicios distribuidos.
C	Para el usuario, la CN es facilitada por proveedores externos, mientras los clústeres AD son comprados y mantenidos por la empresa que los utiliza.
D	Los clústeres AD suelen utilizar un modelo de servicio IaaS y los sistemas de CN proporcionan un modelo de servicio SaaS.

3. Uno de los problemas fundamentales que deben resolver los sistemas distribuidos es la gestión de defectos y fallos. Para ello, se debe...

A	Enviar los mensajes una sola vez. De otra manera, aquellos agentes que quieran atacar nuestro sistema recogerían información relevante sobre nuestros servicios.
B	Evitar mecanismos de detección de fallos, pues esos mecanismos necesitan muchos recursos y la detección no suele ser fiable.
C	Replicar los componentes de los servicios para garantizar que al menos una instancia esté disponible.
D	Todas las anteriores.

4. El principal objetivo del "middleware" es:

A	Mejorar la seguridad del sistema.
B	Mejorar la escalabilidad del sistema.
C	Ocultar y resolver múltiples problemas de comunicación que pueden surgir entre los componentes de un servicio.
D	Gestionar los fallos.

5. Considerando este programa:

```
// Files.js
const fs = require('fs');
if (process.argv.length<3) {
  console.error('More file names are needed!!');
  process.exit();
}
function handler(name) {
  return function(err,data) {
    if (err) console.error(err);
    else console.log('File '+name+': '+data.length+' bytes.');
  }
}
var files = process.argv.slice(2);
var i=-1;
do {
  i++;
  fs.readFile(files[i], 'utf-8', handler(files[i]))
} while (i<files.length-1);
console.log('We have processed '+files.length+' files.');
```

Se ha llegado a ejecutar el programa con esta orden: "node Files A B". A es un fichero de texto de 234781 bytes y B otro fichero de texto con 430 bytes. ¿Cuál es la salida mostrada en esa ejecución?

A	More file names are needed.
B	File A: 234781 bytes. File B: 430 bytes. We have processed 2 files.
C	"We have processed 2 files." seguido por una línea por fichero, mostrando su nombre y tamaño.
D	No llega a mostrarse nada porque el proceso aborta en su línea 8 sin mostrar ningún mensaje.

6. En el programa de la cuestión anterior, la siguiente afirmación es cierta:

A	Utiliza una promesa para procesar cada fichero.
B	Aborta su ejecución en el cuerpo de la función "handler" porque las funciones no pueden retornar funciones.
C	Utiliza la función fs.readFile de manera sincrónica para evitar problemas cuando los nombres y tamaños de los ficheros deban mostrarse.
D	La función "handler" proporciona una clausura para mostrar adecuadamente el nombre de cada fichero.

7. En el algoritmo de exclusión mutua con servidor central, es cierto que:

A	Es un algoritmo altamente disponible, pues todos sus agentes están replicados por omisión.
B	Usa un patrón sincrónico petición-respuesta para obtener el permiso de acceso a la sección crítica y un envío unidireccional asincrónico para liberarla.
C	Usa un patrón unidireccional asincrónico PUSH-PULL para obtener el permiso de acceso a la sección crítica y un PUB-SUB para liberarla.
D	Usa un patrón asincrónico PUB-SUB para obtener el permiso de acceso a la sección crítica y un petición-respuesta sincrónico para liberarla.

8. Considerando estos programas, a ejecutar en un mismo ordenador...

```
// client.js
var zmq=require('zmq');
var rq=zmq.socket('dealer');
rq.connect('tcp://127.0.0.1:8888');
var i=1;  rq.send(''+i);
rq.on('message',function(req,rep){
  console.log("%s %s",req,rep);
  if (i==100) process.exit(1);
  rq.send((++i)+'');
});
```

```
// server.js
var zmq = require('zmq');
var rp = zmq.socket('dealer');
rp.bindSync('tcp://127.0.0.1:8888');
rp.on('message', function(msg) {
  var j = parseInt(msg);
  rp.send([msg,(j*3).toString()]);
});
```

La siguiente afirmación es cierta:

A	En el cliente, cuando se muestra el valor del parámetro “req”, su valor es idéntico al de la variable “i”.
B	El cliente envía 101 peticiones al servidor antes de terminar su ejecución.
C	No puede haber dos o más instancias del cliente en el mismo ordenador. Todas menos la primera abortarían en su intento de conectar con el servidor.
D	Ninguna de las anteriores.

9. Considerando los programas mostrados en la cuestión anterior...

A	El cliente no puede enviar una nueva petición hasta que la respuesta a la anterior sea recibida y procesada.
B	Aunque el patrón de comunicación DEALER-DEALER sea utilizable, en este ejemplo los mensajes no incluyen un delimitador. Sin él, los mensajes no se entregan.
C	Estos programas son inútiles pues no se puede intercomunicar a dos procesos utilizando sockets DEALER en ambos programas.
D	El servidor únicamente puede procesar las peticiones enviadas por un solo cliente.

10. Supongamos que una aplicación distribuida necesita un canal de comunicación unidireccional asincrónico entre dos componentes A y B, enviando los mensajes desde A a B (es decir, A --> B). Para lograr esa comunicación, se puede utilizar este patrón ZeroMQ:

A	A: PULL, B: PUSH.
B	A: REQ, B: REP.
C	A: SUB, B: PUB.
D	A: DEALER, B: DEALER.