Aquest examen conté 20 qüestions d'opció múltiple. En cadascuna d'elles solament una de les seues respostes és correcta. Les contestacions han de presentar-se en una fulla que s'ha entregat a part. Les respostes correctes aporten 0.5 punts a la nota del parcial mentre que les incorrectes resten una dècima.

En la fulla de respostes emplena la casella triada acuradament. Utilitza un llapis o un bolígraf (negre o blau fosc). Es pot usar "Tipp-Ex" o algun corrector similar. En aquest cas, NO INTENTES DIBUIXAR DE NOU LA CASELLA QUE HAGES ESBORRAT.

TEORIA

1. Els sistemes distribuïts...

A	...estan formats generalment per múltiples agents que s'executen concurrentment. Aquests agents mantenen cert estat independent.
B	...no necessiten cap mecanisme de comunicació entre ordinadors.
C	...sempre utilitzen interaccions client-servidor.
D	...no tindran mai condicions de carrera.
E	Totes les anteriors.
F	Cap de les anteriors.

2. La computació en el núvol (*Cloud computing*)...

A	...és un dels paradigmes actuals de prestació de serveis en la computació distribuïda.
B	...té com a objectiu la prestació de serveis de còmput d'una manera escalable i eficient.
C	...segueix un model de "pagament per ús".
D	...usa generalment infraestructures virtualitzades.
E	Totes les anteriors.
F	Cap de les anteriors.

3. En un sistema distribuït, la interacció entre els seus agents...

A	…no s'ha de dur mai a terme. Si interactuaren, el sistema seria concurrent en lloc de distribuït.
B	…es realitza intercanviant missatges o compartint memòria.
C	…es realitza sense compartir memòria. La compartició de memòria està prohibida en els sistemes distribuïts.
D	…s'aconsegueix quan tots els agents residisquen en un mateix ordinador.
E	Totes les anteriors.
F	Cap de les anteriors.

4. El model de programació guarda / acció ...

A	Se segueix en la programació multi-fil, on les seccions crítiques equivalen a les guardes i els fils ("threads") equivalen a les accions.
B	Se segueix en la programació asincrònica (o dirigida per esdeveniments), on els esdeveniments equivalen a les accions i les funcions "callback" dels esdeveniments equivalen a les guardes.
C	Se segueix en la programació multi-fil, on els fils ("threads") equivalen a guardes que s'activen i se suspenen, i les operacions en les seccions crítiques equivalen a les accions.
D	Se segueix en la programació asincrònica (o dirigida per esdeveniments), on els esdeveniments equivalen a les guardes i les funcions "callback" dels esdeveniments equivalen a les accions.
E	Totes les anteriors.
F	Cap de les anteriors.

5. Algunes característiques rellevants dels models de sistemes distribuïts són...

A	Se centren en les principals propietats del comportament del sistema.
B	Faciliten una bona eina per a raonar sobre la correcció dels algorismes i protocols basats en ells.
C	El seu alt nivell d'abstracció.
D	Faciliten una base per a discutir sobre la impossibilitat de resoldre problemes en certs sistemes distribuïts (p. ex., el consens en sistemes asincrònics).
E	Totes les anteriors.
F	Cap de les anteriors.

6. Els elements a considerar en un model de sistema poden ser...

A	L'arquitectura de l'equip, el sistema operatiu, el middleware i el llenguatge de programació.
B	Processos, esdeveniments, aspectes de comunicació, fallades, gestió del temps i nivell de sincronia.
C	Nivell físic, nivell d'enllaç, nivell de xarxa, nivell de transport i nivell d'aplicació.
D	Sistema gestor de bases de dades, middleware i interfície d'usuari.
E	Totes les anteriors.
F	Cap de les anteriors.

7. En la programació de sistemes distribuïts, l'ús del middleware és aconsellable perquè...

A	Introdueix múltiples transparències, ocultant detalls de baix nivell i oferint una interfície uniforme.
B	Té una implantació senzilla, i poca complexitat en els elements manejats.
C	Proporciona una operativa estandarditzada, comprensible i ben definida.
D	Facilita la interoperabilitat, la interacció amb productes de terceres parts.
E	Totes les anteriors.
F	Cap de les anteriors.

8. Els problemes que trobem en els sistemes distribuïts orientats a objectes són:

A	Tots els objectes semblen ser locals i això pot generar llargs intervals per a completar la seua invocació en cas que siguen remots.
B	Els objectes mantenen estat i aquest estat es compartirà entre els agents que invoquen els seus mètodes. Això pot provocar problemes de consistència.
C	El seu estat compartit necessita mecanismes de control de concurrència. Això pot ocasionar bloquejos, evitant que els sistemes siguen escalables.
D	Els seus mecanismes d'invocació faciliten una alta transparència d'ubicació. Això exigeix protocols de recuperació complexos per a gestionar les fallades.
E	Totes les anteriors.
F	Cap de les anteriors.

SEMINARIS

9. Considere's el següent programa (incomplet) escrit en Node:

```
function logaritme(x,b) { return Math.log(x)/Math.log(b) }
function logBase ... // a completar
log2 = logBase(2);
log8 = logBase(8);
console.log("Logarithm base 2 of 1024 = " + log2(1024));
console.log("Logarithm base 8 of 4096 = " + log8(4096));
```

Quina implementació de la funció `logBase` seria correcta?

A	`function logBase(b) { return logaritme(x,b) }`

B	`function logBase(b) {`
	`  return function(x) { return logaritme(x,b) }`
	`}`

C	`function logBase(x) {`
	`  return function(b) { logaritme(x,b) }`
	`}`

D	`function logBase(x) {`
	`  return function(b) { return logaritme(x,b) }`
	`}`

E	Totes les anteriors.
F	Cap de les anteriors.

10. Considere's el següent programa escrit en Node:

```
var fruits = ["Banana", "Orange", "Lemon", "Apple"];
var numbers = [7, 3, "Cloud", 9];
var funcs = [function(x) {return 2*numbers[x]},
             function(x) {return fruits[x]}];
var s = "";
for (var i=0; i<2; i++)
  for (var j=0; j<5; j=j+2)
     s += funcs[i](j) + ", ";
console.log(s);
```

En executar-ho, l'eixida que es mostrarà en consola serà:

A	14, 6, NaN, Banana, Orange, Lemon,
B	14, NaN, NaN, Banana, Lemon, undefined,
C	14, 2Cloud, undefined, Banana, Lemon, undefined,
D	Banana, 14, Lemon, NaN, undefined,
E	No es mostraria res, excepte un missatge d'error indicant que l'array `numbers` està mal definit, per contenir valors de diferents tipus.
F	Cap de les anteriors.

11. **Considere's la següent funció escrita en Node:**

```
function f(x,y) {
   x = x || 'taronja'; y = y || 98;
   console.log('x='+x+' y='+y);
}
```

Indique quin seria l'eixida que es mostrarà en consola si s'executa:

```
f(36);          f(undefined,'poma');    f(45,0,67);
```

A	x=36 y=98	x=undefined y=poma	x=45 y=0
B	x=36 y=98	x=taronja y=poma	x=45 y=0
C	x=36 y=98	x=taronja y=poma	x=45 y=98
D	x=36 y=36	x=taronja y=poma	x=45 y=67
E	No es mostraria res, excepte missatges d'error ja que hi ha invocacions incorrectes (pel seu nombre d'arguments) de la funció f.		
F	Cap de les anteriors.		

12. **Considere's el següent programa escrit en Node:**

```
var eve = new (require('events')).EventEmitter;
var s = "print";
var n = 0;
var handler = setInterval( function(){eve.emit(s);}, 1000 );
eve.on(s, function() {
   if ( n < 2 ) console.log("Event", s, ++n, "times.");
   else clearInterval(handler);
});
```

Si s'executa aquest programa indique, en relació a l'eixida que es mostrarà en consola i al temps d'execució, quina de les següents opcions és la correcta:

A	Event print 1 times. Event print 2 times.	I conclouria després de 3 segons.
B	Event print 1 times. Event print 2 times. Event print 3 times.	I conclouria després de 4 segons.
C	Event print 1 times. Event print 2 times. …	I no conclouria. Cada segon, mostraria una nova línia amb el nombre incrementat en una unitat.
D	Event print 0 times. Event print 1 times. …	I no conclouria. Cada 10 segons, mostraria una nova línia amb el nombre incrementat en una unitat.
E	No es mostraria res, perquè no està ben definit l'objecte listener.	I no conclouria, ja que s'emetria cíclicament l'esdeveniment “print”.
F	Cap de les anteriors.	

13. Considerant el programa següent...

```
var http = require('http');
var fs = require('fs');
http.createServer(function(request,response) {
  fs.readdir(__dirname, function(err,data) {
    if (err) {
      response.writeHead(404, {'Content-Type':'text/plain'});
      response.end('Unable to read directory '+__dirname);
    } else {
      response.writeHead(200, {'Content-Type':'text/plain'});
      response.write('Directory: ' + __dirname + '\n');
      response.end(data.toString());
    }
  })
}).listen('1337');
```

Seleccione les opcions correctes:

A	Aquest programa genera una excepció i avorta en cas de no poder llegir el contingut del directori actual.
B	Aquest programa és un servidor web que respon amb el nom i llista de fitxers en el directori actual.
C	Aquest programa no funciona perquè no ha declarat la variable "__dirname" i no ha importat el mòdul 'process' on està definida.
D	Aquest programa no funciona perquè 'data' és un vector de noms de fitxer i els vectors no poden ser transformats en cadenes.
E	Totes les anteriors.
F	Cap de les anteriors.

14. Alguns problemes de l'algorisme d'exclusió mútua amb servidor central són...

A	No compleix la seua condició de vivacitat.
B	No compleix la seua condició de seguretat.
C	Necessita més missatges que els altres algorismes vists al Seminari 2 per a resoldre el problema d'exclusió mútua.
D	És fràgil en situacions de fallada. El servidor central és un punt únic de fallada.
E	Totes les anteriors.
F	Cap de les anteriors.

15. Els algorismes d'elecció de líder...

A	...són un subconjunt dels algorismes de consens.
B	...necessiten que tots els processos tinguen un identificador diferent.
C	...usen un criteri determinista per a triar al líder.
D	...exigeixen que es trie solament a un procés.
E	Totes les anteriors.
F	Cap de les anteriors.

16. Suposem que es necessita implantar un servei de xat utilitzant node.js i ØMQ. El servidor difon els missatges dels usuaris i no ha de suspendre's mai tractant d'enviar un missatge (de qualsevol tipus). Els programes clients envien els missatges dels usuaris al servidor, esperen els missatges reexpedits pel servidor i informen al servidor quan un usuari s'incorpora o abandona el sistema. Per a implantar aquest servei de xat...

A	El servidor ha d'usar un socket PULL i un altre REP per a interactuar amb els clients.
B	El servidor ha d'usar un socket SUB i un altre REQ per a interactuar amb els clients.
C	El servidor ha d'usar un socket PUB i un altre PULL per a interactuar amb els clients.
D	El servidor ha d'usar un socket REP i un altre SUB per a interactuar amb els clients.
E	Totes les anteriors.
F	Cap de les anteriors.

17. Assumim que hem implantat un servei suportat per múltiples (p. ex., 10) processos servidors situats en ordinadors diferents. Aquests servidors utilitzen sockets REP i els seus clients usen sockets REQ. Si construïm un broker amb un socket ROUTER com a *front-end* i un socket DEALER com a *back-end* (i per a tots dos es realitza un bind()), llavors...

A	Els clients no necessiten conèixer quants processos servidors tenim.
B	Els clients no necessiten conèixer les adreces i ports de cada procés servidor.
C	La quantitat de processos servidors pot variar dinàmicament. Ells han de connectar-se al socket *back-end* perquè el broker puga utilitzar-los.
D	El broker no ha de modificar cap segment dels missatges per a propagar-los del *front-end* al *back-end* i del *back-end* al *front-end*.
E	Totes les anteriors.
F	Cap de les anteriors.

Per a contestar a las següents 2 qüestions (les 18 i 19), consideren-se els següents programes Node amb ØMQ. Un servidor (*server.js*):

```
var zmq = require('zmq')
var rep = zmq.socket('rep')
rep.bindSync('tcp://127.0.0.1:'+process.argv[2])
var n = 0
rep.on('message', function(msg) {
  console.log('Request: ' + msg)
  rep.send('World ' + ++n)
})
```

I un client (*client.js*):

```
var zmq = require('zmq')
var req = zmq.socket('req')
req.connect('tcp://127.0.0.1:'+process.argv[2])
req.connect('tcp://127.0.0.1:'+process.argv[3])
var n = 0
setInterval( function() { req.send('Hello ' + ++n) }, 100 )
req.on('message', function(msg) {
  console.log('Response: ' + msg)
})
```

18. Consideren-se els anteriors programes Node amb ØMQ (*server.js* i *client.js*). Si, en 3 terminals, s'executen 2 servidors i 1 client mitjançant:

```
node server 8001
node server 8002
node client 8001 8002
```

Les primeres línies que es mostraran en les terminals *dels servidors* seran:

A	En una terminal:
	`Request: Hello 1`
	`Request: Hello 3`
		I en l'altra terminal:
		`Request: Hello 2`
		`Request: Hello 4`

B	En ambdues terminals:
	`Request: Hello 1`
	`Request: Hello 2`	

C	En ambdues terminals (sent x, y, z… nombres tals que x < y < z < …):
	`Request: Hello x`
	`Request: Hello y`
	`Request: Hello z`
	`...`	

D	En una terminal:
	`Request: Hello 1`
	`Request: Hello 2`
		I en l'altra terminal:
		`Request: Hello 3`
		`Request: Hello 4`

E	No es mostraria res, atès que el client no sabria a quin dels servidors enviar les seues peticions. (Per a un funcionament correcte, el client hauria de connectar-se a un socket ROUTER).	
F	Cap de les anteriors.	

19. Consideren-se els mateixos programes, i el mateix escenari d'execució, de la qüestió anterior. Les primeres línies que es mostraran en la terminal *del client* seran:

A	`Response: World 1`
	`Response: World 2`
	`Response: World 3`
	`Response: World 4`

B	`Response: World 1`
	`Response: World 1`
	`Response: World 2`
	`Response: World 2`

C	`Response: World 1`
	`Response: World 3`
	`Response: World 5`
	`Response: World 7`

D	`Response: World 1`
	`Response: World 3`
	`Response: World 2`
	`Response: World 4`

E	No es mostraria res, atès que, com el client no sabria a quin dels servidors enviar les seues peticions, cap dels servidors podria enviar respostes.
F	Cap de les anteriors.

20. **Consideren-se els següents programes Node amb ØMQ. Un publicador:**

```
var zmq = require('zmq')
var pub = zmq.socket('pub').bindSync('tcp://*:5555')
var count = 0
setInterval(function() {
  pub.send('PRG ' + count++)
  pub.send('TSR ' + count++)
}, 1000)
```

I un subscriptor:

```
var zmq = require('zmq')
var sub = zmq.socket('sub')
sub.connect('tcp://localhost:5555')
sub.subscribe('TSR')
sub.on('message', function(msg) {
  console.log('Received: ' + msg)
})
```

Si s'executara en primer lloc el publicador i, tres segons després, el subscriptor... Les primeres línies de l'eixida que es mostraran en la terminal del subscriptor seran:

A	`Received: TSR 1`
	`Received: TSR 2`
	`Received: TSR 3`
	…

B	`Received: TSR 1`
	`Received: TSR 3`
	`Received: TSR 5`
	…

C	`Received: PRG 4`
	`Received: TSR 5`
	`Received: PRG 6`
	…

D	`Received: TSR 5`
	`Received: TSR 7`
	`Received: TSR 9`
	…

E	`Received: PRG 0`
	`Received: TSR 1`
	`Received: PRG 2`
	…

F	Cap de les anteriors.