This exam consists of 20 multiple choice questions. In every case only one of the answers is correct. You should 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: -0,1 points.

In the answer sheet, fill carefully your chosen slot. To this end, use a dark pen or pencil. You may use “Tipp-Ex” or any other correction fluid to change your answer. In that case, please do not try to rebuild the empty slot. Leave it without any surrounding box.

THEORY

1. Distributed systems...

A	...usually consist of multiple agents that are executed concurrently. Those agents maintain some independent state.
B	...do not need any inter-computer communication mechanism.
C	...always use a client-server interaction mechanism.
D	...always prevent race conditions from appearing.
E	All the above.
F	None of the above.

2. Cloud computing...

A	...is one of the current service provision paradigms in distributed computing.
B	...aims at providing software services in a scalable and efficient way.
C	...follows a “pay per use” model.
D	...uses virtualised infrastructures in the common case.
E	All the above.
F	None of the above.

3. In a distributed system, its agents interact…

A	…in no way. If they interact, then the system is concurrent, but not distributed.
B	…exchanging messages or sharing memory.
C	…in any way that does not imply any sharing of memory. Memory sharing is forbidden in distributed systems.
D	…sharing all agents the same computer.
E	All the above.
F	None of the above.

4. The guard / action programming model is used in...

A	Multi-threaded programmes, where all critical sections are equivalent to guards and the threads are equivalent to actions.
B	Asynchronous programming, where its events are actions and its *callback* functions (i.e., event listeners) are guards.
C	Multi-threaded programmes, where its threads are guards and critical sections are actions.
D	Asynchronous programming, where its events are guards and its event listeners are actions.
E	All the above.
F	None of the above.

5. Some expected features in distributed system models are…

A	They are centred on the main properties of the system behaviour.
B	They provide a good basis to reason about the correctness of the algorithms and protocols based on them.
C	A high level of abstraction.
D	They provide a basis for discussing about the impossibility of solving problems in some kinds of distributed systems; e.g., consensus in asynchronous systems.
E	All the above.
F	None of the above.

6. **The elements of a system model can be...**

A	Computer architecture, operating system, middleware, and programming language.
B	Processes, events, communication aspects, failures, time management and level of synchrony.
C	Physical layer, data link layer, network layer, transport layer and application layer.
D	Database management system, middleware and user interface.
E	All the above.
F	None of the above.

7. **In distributed computing, the use of middleware is recommended because...**

A	It introduces multiple transparencies, hiding low-level details and providing a uniform interface.
B	It facilitates the implementation of programmes and reduces the complexity of the elements being handled.
C	It provides a standardised, well understood and well defined way of doing things.
D	It improves interoperability; i.e., the interaction with products from third parties.
E	All the above.
F	None of the above.

8. **Problems in distributed object-oriented systems:**

A	All objects seem to be local to their caller and this may hide lengthy invocation intervals.
B	Objects maintain state and that state is shared by all agents that are invoking their methods. This may lead to consistency problems.
C	Their shared state needs concurrency control mechanisms. Those mechanisms lead to blocking intervals that prevent these systems from being scalable.
D	Their invocation mechanisms provide a high degree of location transparency. This fact demands complex recovery protocols in order to deal with failures.
E	All the above.
F	None of the above.

SEMINARS

9. Considering this (incomplete) node programme:

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

Which implementation of `logBase()` is correct?

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

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

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

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

E	All the above.
F	None of the above.

10. Considering this node programme:

```
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);
```

Its output (once it is run) will be:

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 output. Only an error message saying that the “numbers” array has been incorrectly defined, since it holds values of different types.
F	None of the above.

11. Considering the following node function:

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

Please choose will be its output when it is invoked in the following ways:

	`f(36);`	`f(undefined,'apple');`	`f(45,0,67);`
A	x=36 y=98	x=undefined y=apple	x=45 y=0
B	x=36 y=98	x=orange y=apple	x=45 y=0
C	x=36 y=98	x=orange y=apple	x=45 y=98
D	x=36 y=36	x=orange y=apple	x=45 y=67
E	Nothing is shown besides some error messages since there are incorrect calls (due to an incorrect number of arguments) to function f.		
F	None of the above.		

12. Considering the following node programme...

```
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);
});
```

Please select the correct choice regarding programme output (first part) and execution time (second part):

A	Event print 1 times. Event print 2 times.	The programme is terminated in 3 seconds.
B	Event print 1 times. Event print 2 times. Event print 3 times.	The programme is terminated in 4 seconds.
C	Event print 1 times. Event print 2 times. …	The programme is never terminated. Each second it shows a new line with an increased number.
D	Event print 0 times. Event print 1 times. …	The programme is never terminated. Every 10 seconds it shows a new line with an increased number.
E	Nothing is shown since the listener has not been defined correctly.	The programme is never terminated, since it emits cyclically the “print” event.
F	None of the above.	

13. Considering this node.js programme...

```
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');
```

The following sentences are true:

A	This programme raises an exception and aborts when it is unable to read the contents of the current directory.
B	This programme is a web server that replies with the name and list of files in its current directory.
C	This programme does not work since it has not declared variable "__dirname" and it has not imported the 'process' module where such variable is defined.
D	This programme does not work since 'data' is an array of file names and arrays cannot be cast into strings.
E	All the above.
F	None of the above.

14. Some problems of the central server algorithm for mutual exclusion are...

A	It does not respect its liveness condition.
B	It does not respect its safety condition.
C	It requires more messages than the other algorithms seen in Seminar 2 for solving the mutual exclusion problem.
D	It is fragile when failures arise. The central server is a single point of failure.
E	All the above.
F	None of the above.

15. Leader election algorithms...

A	...are a subset of consensus algorithms.
B	...require that all processes have a different identifier.
C	...use a deterministic criterion to select the leader.
D	...require that only one process is elected.
E	All the above.
F	None of the above.

16. We want to implement a chat service using node.js and ØMQ. The server multicasts the user messages and should never block trying to send a message (of any kind). The client programmes send user messages to the server, wait for user messages forwarded by the server and report to the server when a user enters or abandons the system. In order to implement this chat service...

A	The server must use a PULL and a REP socket to interact with clients.
B	The server must use a SUB and a REQ socket to interact with clients.
C	The server must use a PUB and a PULL socket to interact with clients.
D	The server must use a REP and a SUB socket to interact with clients.
E	All the above.
F	None of the above.

17. Let us assume that we have implemented a service supported by several (e.g., 10) server processes placed in different computers. Those servers use REP sockets and their clients use REQ sockets. If we build an intermediate broker with a front-end ROUTER socket and a back-end DEALER socket (both bound to static URLs), then...

A	Clients do not need to know how many servers exist.
B	Clients do not need to know the addresses and ports of each server process.
C	The amount of server processes may be dynamically changed. They need to connect to the back-end socket in order to be available for the broker.
D	The broker does not need to modify any message segment in order to correctly forward messages from front-end to back-end and from back-end to front-end.
E	All the above.
F	None of the above.

In order to answer the following two questions (i.e., 18 and 19), please consider the following node programmes with ZeroMQ. A server (*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)
})
```

...and a 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. Considering the programmes (*server.js* and *client.js*) shown previously... Let us assume that 2 servers and 1 client have been started in three consoles using:

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

The first lines printed in the server consoles will be:

A	In a console:
	`Request: Hello 1`
	`Request: Hello 3`
		And in the other:
		`Request: Hello 2`
		`Request: Hello 4`

B	In both consoles:
	`Request: Hello 1`
	`Request: Hello 2`	

C	In both consoles: (being x, y, z… numbers such that x < y < z < …):
	`Request: Hello x`
	`Request: Hello y`
	`Request: Hello z`
	`...`	

D	In a console:
	`Request: Hello 1`
	`Request: Hello 2`
		And in the other:
		`Request: Hello 3`
		`Request: Hello 4`

E	Nothing is printed since the client is unable to decide to which server it must send its requests. (To fix this problem, the client should be connected to a ROUTER socket.)	
F	None of the above.	

19. Considering again those two programmes and the same execution scenario than in the previous question... The first lines printed in the client console will be:

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	Nothing is printed. Since the client cannot send its requests, no server will answer.
F	None of the above.

20. Considering the following programmes... A publisher:

```
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)
```

And a subscriber:

```
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)
})
```

Assuming that the publisher has been launched first and, three seconds later, we have started the subscriber... The first lines shown in the subscriber console will be:

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	None of the above.