L'examen consta de 20 qüestions d'opción múltiple. En cada qüestió hi ha una única resposta correcta. Les respostes han de proporcionar-se en una fulla SEPARADA que s'ha facilitat al costat d'aquest enunciat.

Totes les qüestions tenen el mateix valor. Si es responen correctament, aporten 0.5 punts a la nota obtinguda. Si la resposta és negativa, l'aportació és negativa i equivalent a 1/5 del valor correcte; és a dir, -0.1 punts. En cas de dubte es recomana deixar la resposta en blanc.

La durada d'aquesta part de l'examen és 1 hora.

1. L'ordre "`git push origin master`"

A	Porta tots els commits del dipòsit remot "`origin`" al nostre dipòsit local.
B	Esborra el directori de treball actual.
C	Sempre copia els commit locals en el dipòsit remot.
D	Quan té èxit, deixa la branca del dipòsit remot master apuntant al mateix commit que la branca master local.
E	Totes les anteriors.
F	Cap de les anteriors.

2. Les dos ordres "`git pull`" i "`git fetch`"

A	Són equivalents.
B	Porten tots els commits del dipòsit remot al dipòsit local.
C	Deixen la branca master local apuntant al mateix commit que la branca master remota.
D	Modifiquen el directori de treball.
E	Totes les anteriors.
F	Cap de les anteriors.

3. GitLab és …

A	Un servidor concret dins de la UPV.
B	El nom d'un dipòsit.
C	Una forma de confirmar (*commit*) canvis en un dipòsit.
D	Una versió experimental de Git.
E	Totes les anteriors.
F	Cap de les anteriors.

4. En el flux de treball per a la fase de desenvolupament que s'ha sol·licitat utilitzar en GitLab...

A	Cada desenvolupador de l'equip ha de tenir un dipòsit remot privat.
B	Hi ha un dipòsit canònic que cada membre de l'equip pot llegir i que és diferent dels dipòsits remots privats.
C	Solament el propietari pot escriure en el seu dipòsit remot privat.
D	Solament un membre de l'equip ha d'escriure en el dipòsit canònic.
E	Totes les anteriors.
F	Cap de les anteriors.

5. En un equip TSR, cada dipòsit privat remot...

A	Ha de ser accesible només pel seu propietari.
B	Ha de ser accessible en mode escriptura pel seu propietari.
C	Ha de ser accessible en mode escriptura per l'integrador de l'equip.
D	Ha de ser accessible en mode escriptura pel líder de l'equip.
E	Totes les anteriors.
F	Cap de les anteriors.

6. Un dipòsit Git local…

A	Ha d'estar associat a un dipòsit Git remot.
B	Només pot associar-se a un únic dipòsit remot.
C	Pot clonar-se sobre un altre dipòsit local.
D	No pot ser modificat.
E	Totes les anteriors.
F	Cap de les anteriors.

7. Seleccione l'ordre que crea un dipòsit Git local buit...

A	“`git init; touch README; git commit`”
B	“`git clone git:init`”
C	“`git init origin master`”
D	“`git init -o origin master`”
E	Totes les anteriors.
F	Cap de les anteriors.

8. Assumisca la següent situació: El dipòsit R (la URL del qual és R_{url}) pot ser accedit per dos desenvolupadors D1 i D2. La branca "master" de R apunta a un commit C que conté el fitxer "`foo.js`", amb una sola línea, "`var fs = require('fs')`". Cada desenvolupador executa "`git clone` R_{url}`; cd R;`" en el seu ordinador. D1 edita "`foo.js`" i afig la línia "`fs.readFileSync('myfile')`". D2 també edita "`foo.js`" però afig la línia "`fs.readFileSync('nofile')`". Tots dos fan el commit dels seus canvis, en diferents moments, obtenint els commits C1 i C2, respectivament. Llavors ells executen "`git push`".
Podem afirmar que…

A	No canvia res en R.
B	La branca "master" de R apuntarà a C1.
C	La branca "master" de R apuntarà a C2.
D	Un o més desenvolupadors obtindran un error quan executen "`git push`".
E	Totes les anteriors.
F	Cap de les anteriors.

Considere el següent fragment 1:

```
01:   var net = require('net');
02:   var server = net.createServer(  function(c) {
03:      console.log('server connected');
04:      c.write('World');
05:      c.end();
06:   });
07:
08:   server.listen(8000, function() {
09:     console.log('server bound');
10:   });
11:
```

9. Assumisca que el procés P1 executa el codi del fragment 1. Si assumim que P2 és un altre procés iniciat al mateix ordinador que P1 i que P2 es connecta al port 8000, aleshores…

A	P1 envia la resposta a P2 després de rebre el missatge de petició.
B	Quan P1 es fica a escoltar imprimeix el missatge 'server connected'.
C	P2 podria rebre el missatge 'World' abans d'enviar cap informació a P1.
D	Si P2 tanca el seu socket, això no implica que P1 es desconnecte.
E	Totes les anteriors.
F	Cap de les anteriors.

Considere el següent fragment 2:

```
01:  var net = require('net');
02:
03:  var server_port = process.argv[2];
04:  var text        = process.argv[3].toString();
05:  var client      = net.connect({port: parseInt(server_port}, function() {
06:     console.log('client connected');
07:     // This will be echoed by the server.
08:     client.write(text);
09:  });
10:  client.on('data', function(data) {
11:    console.log(data.toString());
12:    client.end();
13:  });
14:
```

10. Suposem que el procés P1 executa aquest fragment 2. Siga P2 un procés servidor que escolta peticions al port 8000. Aleshores …

A	P1 trau un missatge per pantalla quan P2 tanca el socket.
B	La connexió es produirà encara que P2 no s'execute en la mateixa màquina que P1.
C	Si el servidor no està en marxa el missatge contingut en la variable 'text' quedarà a l'espera al buffer d'eixida fins que el servidor estiga en marxa.
D	Es podria invocar P1 des de la línia d'ordres de Linux com: $ node client 127.0.0.1 8000 hello
E	Totes les anteriores.
F	Cap de les anteriors.

Considere el fragment 3 següent:

```
01:  var net = require('net');
02:
03:  var LOCAL_PORT  = 8000;
04:  var LOCAL_IP  = '127.0.0.1';
05:  var REMOTE_PORT = 80;
06:  var REMOTE_IP = '158.42.156.2';
07:
08:  var server = net.createServer(function (socket) {
09:      socket.on('data', function (msg) {
10:         var serviceSocket = new net.Socket();
11:         serviceSocket.connect(parseInt(REMOTE_PORT), REMOTE_IP, function (){
12:             serviceSocket.write(msg);
13:         });
14:         servicesocket.on('data', function (data) {
15:             socket.write(data);
16:         });
17:         console.log("Client connected");
18:      });
19:  }).listen(LOCAL_PORT, LOCAL_IP);
20:  console.log("TCP server accepting connection on port: " + LOCAL_PORT);
```

Assumisca que aquest codi s'executa per a generar el procés P1. Les dades dels clients connectats al port 8000 s'assumeix que arriben en blocs a través de la connexió que aquests estableixen. Cada bloc genera un esdeveniment "data" en el socket que manté la connexió.

Assumisca un procés P2 iniciat després de P1 en l'ordinador de P1 i que també es connecta al port local 8000. Responga les tres qüestions següents:

11. Quan P2 es connecta al port local 8000, P1 …

A	… es connecta a REMOTE_IP.
B	… imprimeix '`Client connected`' en la consola.
C	… comença a esperar dades enviades des de REMOTE_IP.
D	… deixa d'acceptar noves connexions.
E	Totes les anteriors.
F	Cap de les anteriors.

12. Assumim que P2 envia dos blocs grans de dades en seqüència, D1 i després D2, a través de la seua connexió amb P1.
Llavors, en absència de fallades,...

A	D2 sempre arriba a REMOTE_IP després de D1.
B	D1 sempre arriba a REMOTE_IP després de D2.
C	Solament D1 es transmet a REMOTE_IP.
D	Solament D2 es transmet a REMOTE_IP.
E	Totes les anteriors.
F	Cap de les anteriors.

13. Assumim que el servidor en REMOTE_IP només poguera manejar una conexión alhora.
Llavors, si P2 enviara dos blocs de dades...

A	El servidor obtindria dos blocs de dades.
B	El servidor acabaria després de rebre el primer bloc de dades.
C	El servidor rep un sol bloc de dades.
D	El proxy es tanca després de manejar el primer bloc de dades.
E	Totes les anteriors.
F	Cap de les anteriors.

Considere el fragment 4 següent:

```
01:   var net = require('net');
02:
03:   var LOCAL_PORT  = 8000;
04:   var LOCAL_IP  = '127.0.0.1';
05:   var REMOTE_PORT = 80;
06:   var REMOTE_IP = '158.42.156.2';
07:
08:   var server = net.createServer(function (socket) {
09:       console.log("Client connected");
```

```
10:       var serviceSocket = new net.Socket();
11:       serviceSocket.connect(parseInt(REMOTE_PORT), REMOTE_IP, function (){
12:          socket.on('data', function (msg) {
13:              serviceSocket.write(msg);
14:          });
15:          servicesocket.on('data', function (data) {
16:              socket.write(data);
17:          });
18:       });
19:   }).listen(LOCAL_PORT, LOCAL_IP);
20:   console.log("TCP server accepting connection on port: " + LOCAL_PORT);
```

Assumisca que P1 executa aquest codi en comptes del fragment 3, sent la resta idèntica al descrit per al fragment 3. Conteste les tres qüestions següents:

14. Quan P2 es connecta al port local 8000, P1 …

A	… es connecta a REMOTE_IP.
B	… imprimeix '`Client connected`' en la consola.
C	… crea com a màxim una connexió a REMOTE_IP per cada connexió client.
D	… espera dades des de REMOTE_IP tan aviat com establisca una connexió amb REMOTE_IP.
E	Totes les anteriors.
F	Cap de les anteriors.

15. P2 envia dos blocs grans de dades en seqüència, D1 seguit per D2, a través de la seua connexió amb P1.
Llavors, en absència de fallades …

A	D2 sempre arriba a REMOTE_IP després de D1.
B	D1 sempre arriba a REMOTE_IP després de D2.
C	Solament D1 és transmès a REMOTE_IP.
D	Solament D2 és transmès a REMOTE_IP.
E	Totes les anteriors.
F	Cap de les anteriors.

16. Assumisca que el servidor en REMOTE_IP puga gestionar només una connexió alhora.
Llavors, si P2 envia dos blocs de dades …

A	El servidor obté tots dos blocs de dades.
B	El servidor acaba després de rebre el primer bloc de dades.
C	El servidor solament rep un bloc de dades.
D	El proxy es tanca després de manejar el primer bloc de dades.
E	Totes les anteriors.
F	Cap de les anteriors.

17. Tots dos fragments 3 i 4 …

A	Permeten a P1 tancar la connexió a REMOTE_IP quan el client tanca la connexió.
B	Permeten a P1 tancar la connexió amb els seus clients quan REMOTE_IP tanque la seua connexió.
C	Necessiten ser modificats per a capturar l'esdeveniment 'end' en `serviceSocket` i `socket` per a gestionar adequadament el tancament de connexions.
D	No poden ser modificats per a gestionar adequadament el tancament de connexions.
E	Totes les anteriors.
F	Cap de les anteriors.

Considere el següent fragment 5:

```
01:  var net = require('net');
02:
03:  var COMMAND_PORT  = 8000;
04:  var LOCAL_IP  = '127.0.0.1';
05:  var BASEPORT  = COMMAND_PORT + 1;
06:  var remotes = []
07:
08:  function Remote(host, port, localPort) {
09:      this.targetHost = host;
10:      this.targetPort = port;
11:      this.server = net.createServer(function (socket)
12:          WHO.handleClientConnection(socket);
13:      });
14:      if (localPort) {
15:          this.server.listen(localPort, LOCAL_IP);
16:      }
17:  }
18:
19:  Remote.prototype.handleClientConnection = function (socket) {
20:      var serviceSocket = new net.Socket();
21:      serviceSocket.connect(X_SERVER_PORT, X_SERVER_IP, function (){
22:         socket.on('data', function (msg) {
23:             serviceSocket.write(msg);
24:         });
25:         servicesocket.on('data', function (data) {
26:             socket.write(data);
27:         });
28:         X_WHAT.on(X_EVENT, function () {
29:             serviceSocket.end();
30:         });
31:         X_WHICH.on(X_EVENT, function () {
32:             WHAT.end();
33:         });
34:      });
35:  }
36:
37:  Remote.prototype.start = function (localPort) {
38:     this.server.listen(localPort, LOCAL_IP);
39:  }
40:
```

18. **En el fragment 5, la variable** `WHO`

A	Ha de declarar-se en l'àmbit de la funció `Remote`.
B	Ha d'apuntar a l'objecte que s'està construint.
C	No pot ser null.
D	No pot ser undefined.
E	Totes les anteriors.
F	Cap de les anteriors.

19. **En el fragment 5,** `X_WHAT` **ha de substituir-se per...**

A	El socket de la connexió client.
B	El socket de la connexió del servidor.
C	L'objecte Remote.
D	L'objecte servidor.
E	Totes les anteriors.
F	Cap de les anteriors.

20. **En el fragment 5,** `X_SERVER_PORT` **i** `X_SERVER_IP` **...**

A	Es refereixen als atributs `targetHost` i `targetPort` de l'objecte Remote.
B	Han de substituir-se per `this.targetHost` i `this.targetPort`, respectivament.
C	Poden ser idèntics per a diferents objectes Remote.
D	No poden ser null o undefined.
E	Totes les anteriors.
F	Cap de les anteriors.