Ejercicio 1º (3 puntos cuyo desglose por apartados y subapartados se indica)

El área de sistemas de una empresa ha optado por llevar a cabo el diseño de su red interna a partir de la dirección de red **192.168.224.0**, máscara **255.255.254.0**. Con objeto de organizar las interconexiones de los equipos y partiendo de un enrutador (A) conectado a Internet y otro enrutador (B) destinado a estructurar la red en subredes, han considerado que las interfaces de este último enrutador (B) interconecte a las subredes de acuerdo con la tabla siguiente:

Interfaces enrutador B	Subred	Equipos de la subred
GigaEthernet 1	LAN empleados	100 equipos finales
GigaEthernet 2	LAN empresas subcontratadas	60 equipos finales
GigaEthernet 3	Red DMZ	14 servidores
GigaEthernet 4	Conexión punto a punto con enrutador A	-

a) (1 punto) Diseñe el esquema de direccionamiento de la red de tal forma que el tamaño de cada una de las subredes se ajuste lo máximo posible al número de equipos solicitados. Recuerde que en las hojas autocopiativas deben aparecer los cálculos empleados y la reproducción de la tabla siguiente con los resultados finales.

Subred	Dirección de red	Máscara CIDR	Máscara decimal	Rango asignable	Nº máximo hosts	Dirección de broadcast
Empleados		/				
Subcontrata		/				
DMZ		/				
Enlace B-A		/				

b) (1 punto) Considerando que la configuración NAT es la adecuada para los escenarios que se plantean, si la dirección pública asignada por RIPE para la interface exterior del enrutador A es la **95.39.162.7** y en el equipo **192.168.224.200/28** se acaba de ejecutar con éxito el comando: `python3 -m http.server 9000`

b.1) (0,25 puntos) ¿Qué URL (indique la URL completa) habría que solicitar desde un navegador web que se ejecuta en un máquina cuya IP pública es **209.45.3.20** para obtener la respuesta del proceso python versión 3 señalado?

b.2) (0,75 puntos) Explique las transformaciones que se producen en las direcciones IP origen y destino de los paquetes, centrándose en las transformaciones NAT de los enrutadores A y B (por lo tanto, obvie los saltos que se producen en los nodos intermedios de Internet).

c) (1 punto) Si el enrutador B fuese una máquina Linux con múltiples interfaces y tuviese configurado un cortafuegos con IPtables con política DROP para las cadenas de la tabla FILTER.

c.1) (0,5 puntos) ¿Qué comandos IPTables concretos y mínimos tendría que establecer en él para permitir **exclusivamente** que las máquinas de la LAN de empleados (interface eth0) puedan **completar** (solicitud y respuesta) un ping a las máquinas de la LAN de la red de empresa subcontratada (interface eth1), pero denegando el ping en el sentido contrario?

c.2) (0,5 puntos) Tras incorporar los comandos del apartado anterior ¿podrían hacer ping las máquinas de la LAN de empleados al propio cortafuegos? ¿Por qué?

Ejercicio 2º (3 puntos cuyo desglose por apartados y subapartados se indica)

Partiendo de una estructura de directorios tradicional en la que un directorio puede contener varios ficheros y subdirectorios que, a su vez, también pueden estar compuestos por varios ficheros y/o subdirectorios y así sucesivamente, se deben realizar los siguientes métodos/funciones en lenguaje Java o en otros lenguajes que incorporen el paradigma estructurado. **(Ver anexo en pág. 6).**

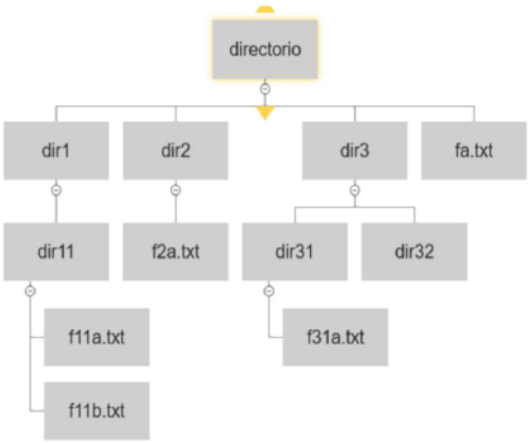
a) (0,5 puntos) Método/función recursivo que reciba como parámetro el nombre del directorio y devuelva el número del nivel al que se encuentra el directorio más profundo de la jerarquía.

```
int numNiveles(String dir);
```

b) (1 punto) Método/función recursivo que reciba como parámetro el nombre del directorio y devuelva una lista con los nombres de los ficheros que estén más profundos en la jerarquía correspondiente.

```
List <String> listadoProfundos(String dir);
```

Por ejemplo, en la estructura siguiente los nodos que empiezan por d son directorios y los que empiezan por f son ficheros. Si se invoca al método/función `numNiveles` con el parámetro "directorio" se debe obtener 2. En el caso del método/función `listadoProfundos` se debe obtener la lista ["directorio/dir1/dir11/f11a.txt", "directorio/dir1/dir11/f11b.txt", "directorio/dir3/dir31/f31a.txt"]



En ambos apartados:

- Se considera que el parámetro directorio está a nivel 0 y dentro del directorio de trabajo actual.
- Se valorará un adecuado control de errores y la justificación de la solución elegida.
- Se podrán codificar métodos auxiliares adicionales al solicitado.
- Se debe elaborar un algoritmo recursivo que recorra todos los nodos del árbol.
- No se hará ninguna suposición sobre el número de niveles del árbol.
- No se podrá basar la solución del ejercicio en el nombre absoluto del nodo, por ejemplo, contando los niveles en base al path absoluto.
- No se podrán utilizar datos, variables, atributos, etc. que no sean locales a las funciones/métodos que se codifican. En definitiva, no se pueden utilizar variables ni atributos globales.

c) (0,5 puntos) Realice las modificaciones necesarias en el método/función `numNiveles` para que mediante gestión de excepciones se trasladen al método/función que lo invoque, los errores que se puedan producir debidos a que el parámetro de entrada no sea válido como directorio para proceder a la ejecución del método `numNiveles`.

d) (1 punto) Realice un juego de pruebas de **cobertura** en JUnit o framework similar para el lenguaje empleado en la solución, que permita comprobar el funcionamiento del método `numNiveles` después de la modificación del apartado anterior. Indique la versión de JUnit utilizada o la identificación y versión del framework empleado.

Ejercicio 3º (4 puntos cuyo desglose por apartados y subapartados se indica)

Parte A: (1,75 puntos)

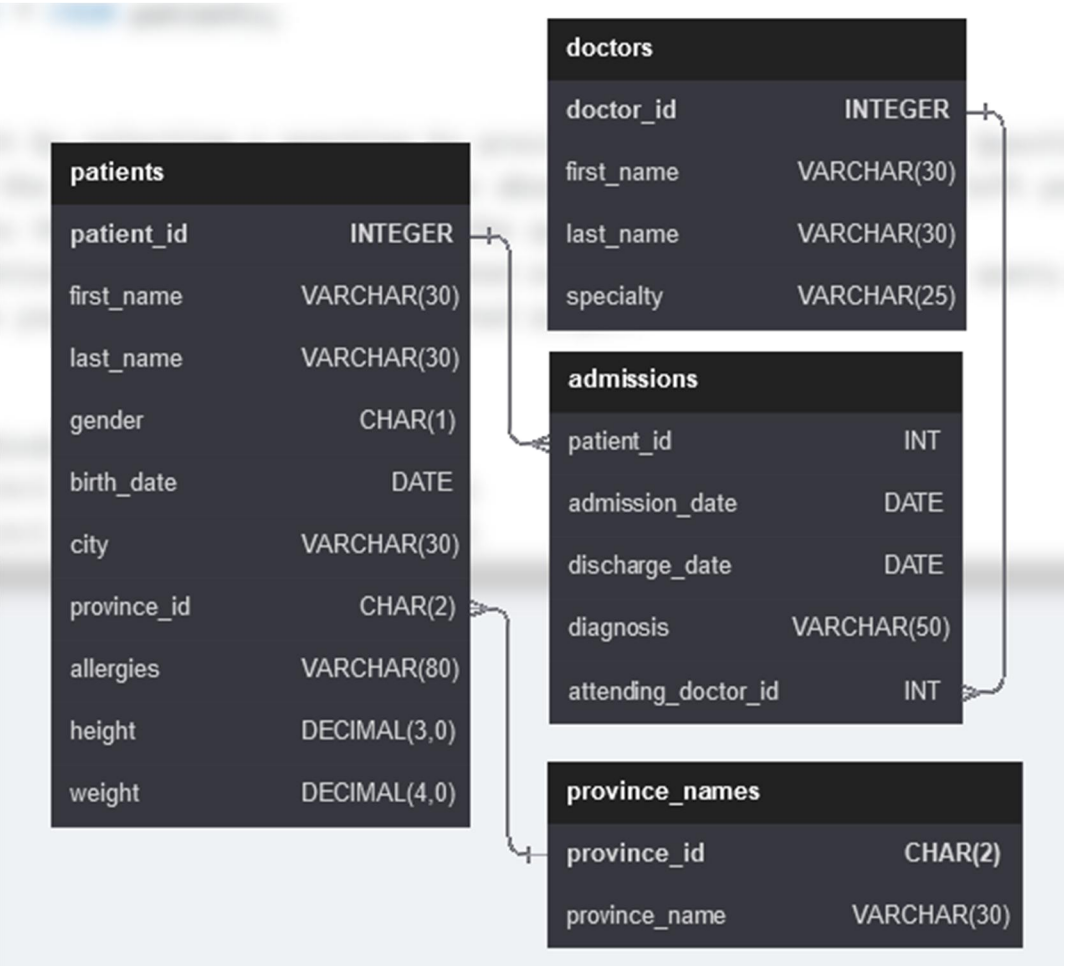
En el ámbito educativo son muy populares algunos esquemas de bases de datos pues permiten poner en práctica los conceptos aprendidos sobre los lenguajes de consulta tipo SQL, QBE, etc. En la figura siguiente aparece representado uno de estos esquemas en el que, de forma muy reducida, se estructura la información de una clínica sanitaria. El esquema incluye las tablas siguientes:

Tabla Pacientes formada por el identificador de paciente (PK), nombre, apellido, género, fecha de nacimiento, ciudad, identificador de provincia (FK), alergias, altura y peso.

Tabla Doctores integrada por el identificador de doctor (PK), nombre, apellido y especialidad.

Tabla de Admisiones formada por el identificador de paciente (FK), fecha de admisión, fecha de alta, diagnóstico e identificador del doctor que asiste al paciente (FK).

Tabla Provincias que contiene el identificador de la provincia (PK) y el nombre de la provincia.



Proponga instrucciones en lenguaje SQL para resolver las consultas siguientes:

a) (0,25 puntos) Muestre en una única fila el número total de pacientes por género

b) (0,25 puntos) Obtenga de cada paciente el nombre completo en una única columna con el formato siguiente: Apellido completamente en mayúsculas seguido de una coma y el nombre completamente en minúsculas. La selección debe aparecer ordenada alfabéticamente de forma descendente por apellido.

c) (0,25 puntos) Obtenga la cantidad de pacientes agregados en grupos formados a través de rangos de peso. Los rangos de peso se calculan a partir de los datos de los pacientes de manera que los pacientes cuyo peso esté entre 100 y 109 ambos inclusive se contabilizarán en un grupo denominado 100. Los que estén entre 110 y 119 en el grupo 110 y así sucesivamente. El resultado debe ordenarse descendentemente por grupo.

d) (0,25 puntos) Muestre el identificador de los pacientes, nombre, apellido y la especialidad de su doctor de aquellos casos diagnosticados con “Epilepsia” y atendidos por la Doctora de nombre “Lisa”.

e) (0,25 puntos) Proyecte exclusivamente el nombre de las provincias que tiene más pacientes de género ‘M’ que de género ‘F’

f) (0,25 puntos) Para que todos los pacientes que han pasado por admisión puedan ver sus documentos médicos se les facilita una contraseña temporal después de su primera admisión. Muestre el identificador del paciente y la contraseña temporal generada, la cual se obtiene concatenando el identificador, la longitud del apellido y el año de nacimiento del paciente.

g) (0,25 puntos) Muestre todos los datos de aquel paciente que cumpla con todos los criterios siguientes: El nombre contiene un letra ‘r’ después de las dos primeras letras, identifica su género como ‘F’, ha nacido en febrero, mayo o diciembre, su peso está entre 60 y 80, su identificador es impar y es de la ciudad de ‘Kingston’.

Parte B: (1,25 puntos)

Dadas las tablas siguientes y la vista definida.

TABLA	CAMPO	TIPO	SIGNIFICADO
dept	deptno	NUMBER (2,0)	Número
	dname	VARCHAR2(14)	Nombre
	loc	VARCHAR2(13)	Localidad
	CONSTRAINT PK_dept PRIMARY KEY (deptno)		

TABLA	CAMPO	TIPO	SIGNIFICADO
emp	empno	NUMBER (4,0)	Número
	ename	VARCHAR2(10)	Apellido
	job	VARCHAR2(9)	Puesto
	mgr	NUMBER(4,0)	Código del jefe
	hiredate	DATE	Fecha de contratación
	sal	NUMBER(7,2)	Sueldo
	comm	NUMBER(7,2)	Comisión
	deptno	NUMBER(2,0)	Código del departamento
	CONSTRAINT PK_emp PRIMARY KEY (empno), CONSTRAINT FK_emp_dept FOREIGN KEY (deptno) REFERENCES dept, CONSTRAINT CK_deptno_NN CHECK (deptno IS NOT NULL)		

```
CREATE OR REPLACE VIEW empl_depar AS
SELECT e.empno, e.ename, e.job, d.dname, d.loc
FROM emp e, dept d
WHERE e.deptno = d.deptno;
```

Se desea codificar un disparador (trigger) que responda al siguiente comportamiento cuando se trate de hacer operaciones DML sobre la vista señalada. **Nota:** *La división en apartados es para reflejar la puntuación asociada, pero hay que codificar todo el comportamiento sobre el mismo disparador.*

a) (0,25 puntos) Si se trata de eliminar un registro, eliminará el empleado correspondiente de la tabla emp.

b) (0,25 puntos) Si se trata de insertar un registro, insertará el empleado correspondiente (usando los campos que se aportan en la sentencia INSERT lanzada sobre la vista) en la tabla emp y:

- Si existe el departamento, le asociará el empleado.
- Si no existe el departamento, lo creará y le asociará el empleado.

c) (0,25 puntos) Si se trata de actualizar el campo dname, lo actualizará para el departamento correspondiente de la tabla dept.

d) (0,25 puntos) Si se trata de actualizar el campo job, lo actualizará para el empleado correspondiente de la tabla emp.

e) (0,25 puntos) En cualquier otro caso, se levantará una excepción con el mensaje "Error en el proceso de actualización sobre la vista empl_depar".

Parte C: (1 punto)

Desarrolle un procedimiento almacenado (o bloque anónimo) que con un cursor definido para la selección de todos los empleados ordenados por número de departamento calcule por cada departamento:

a) (0,25 puntos) Número de empleados del departamento.
b) (0,25 puntos) Media de salarios del departamento.

Esta información se mostrará antes de cambiar de departamento.

Al terminar con todos los departamentos, se mostrará la información totalizada:

c) (0,25 puntos) Número total de empleados.
d) (0,25 puntos) Media total de salarios.

ANEXO – Métodos Java

A continuación, se proporciona la definición de algunos métodos del API Java para gestión de sistemas de ficheros como ayuda complementaria al ejercicio 2º.

```
Interface Path
Path getFileName();
Path getName(int index);
Path getParent();
Boolean isAbsolute();
Class java.nio.file.Paths
static Path get(String first,
String... more);
static Path get(URI uri);
```

```
Class java.io.File
File(File parent, String child);
File(String pathname);
boolean createNewFile();
boolean delete();
boolean exists();
String getName();
String getParent();
String getPath();
File getParentFile();
boolean isDirectory();
boolean isFile();
String[] list();
String[] list(FilenameFilter filter);
File[] listFiles();
File[]
listFiles(FilenameFilter filter);
boolean mkdir();
```

Class java.nio.file.Files

```
static long copy(InputStream in, Path target, CopyOption... options)
static Path createDirectories(Path dir, FileAttribute<?>... attrs)
static Path createFile(Path path, FileAttribute<?>... attrs)
static void delete(Path path)
static boolean isDirectory(Path path, LinkOption... options)
static DirectoryStream<Path> newDirectoryStream(Path dir);
static boolean isRegularFile(Path path, LinkOption... options);
static Stream<Path> list(Path dir);
static boolean    isExecutable(Path path)
static Path  move(Path source, Path target, CopyOption... options)
static DirectoryStream<Path> newDirectoryStream(Path dir, String glob)
static List<String> readAllLines(Path path, Charset cs)   static
List<String> readAllLines(Path path, Charset cs)
```