Ejercicio 1 (2,5 ptos)

Escriba un script llamado ejercicio1.sh que imprima un listado de los números primos en el intervalo [*A*,*B*]. *A* y *B* serán pasados como parámetros. Ejemplo de llamada:

$ ./ejercicio1.sh A B

Ejemplo de ejecución y salida generada para el intervalo [5,14]:

$ ./ejercicio1.sh 5 14
5
7
11
13

NOTA: Puede comprobar si un número es primo analizando la salida de la utilidad **factor** incluida en el paquete GNU *coreutils* (que suponemos estará instalado en su sistema). Dicha utilidad factoriza números enteros. Ejemplo de uso y salida generada de la llamada a la utilidad **factor**:

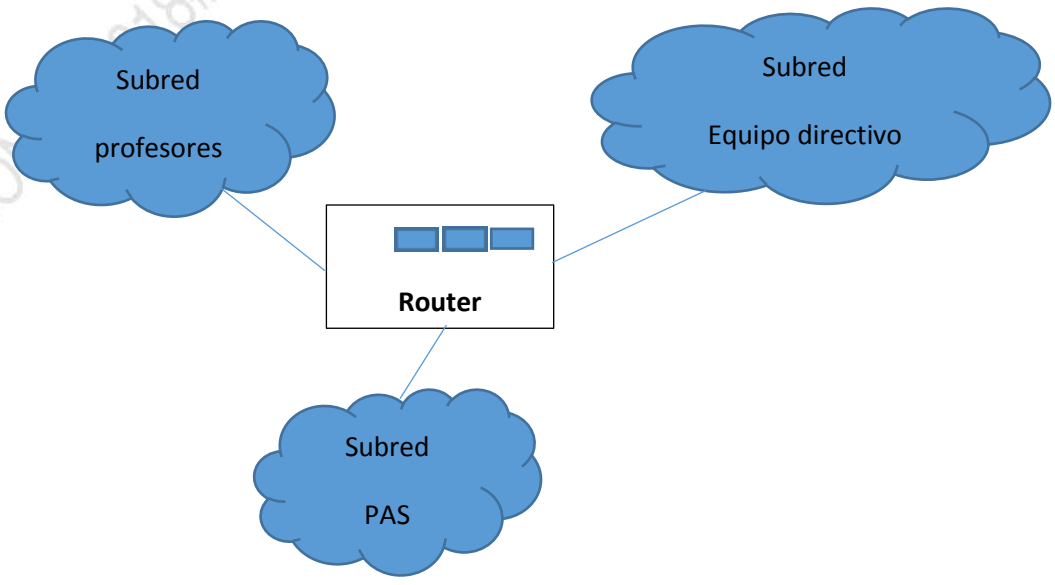
$ factor 2018
2018: 2 1009

Ejercicio 2 (2,5 ptos)

Para efectuar el direccionamiento IP de la red privada de nuestro instituto disponemos de la dirección pública de red 128.42.64.0 y la máscara de red 255.2555.192.0.

a) ¿De qué clase es esta dirección de red y qué porcentaje o fracción de dicha dirección de red cubre esta red privada?

Se desea crear tres subredes dentro de esta red (profesores, equipo directivos y personal de administración y servicios) e interconectarlas a través de un router IP tal y como muestra la siguiente figura:



La red de profesores se quiere dimensionar para 389 direcciones, la red del equipo directivo 123 direcciones y la red de personal de administración y servicios 195 direcciones. Se espera un crecimiento del 5%, por lo que deberemos diseñar las subredes para que, en aquellos casos en que se pueda alcanzar el límite de la subred, se deje espacio equivalente y consecutivo al obtenido sin tener en cuenta el posible crecimiento de dicha subred. Este espacio deberá ser considerado como una subred independiente que no será utilizada por el momento.

b) ¿Cuántas direcciones IP precisará reservar?
c) ¿Qué máscara de red garantiza una subred del tamaño suficiente como para cubrir el espacio de direcciones requerido?
d) ¿Qué máscaras de red precisará cada una de las subredes?

Ejercicio 3 (2,5 ptos)

Desarrolle una mini-aplicación en java que conste de los siguientes scripts:

1.) CreaSesion.java.

En este primer servlet:

- Se crea/recupera una sesión.
- Se resetea su contenido (se elimina cualquier dato que tuviera)
- Se registran 6 variables de sesión con lo siguientes identificadores y tipos:
  - entero: un número entero
  - real: un número real
  - texto: una cadena de texto
  - fecha: una fecha (objeto DateTime)
  - semaforo (array asociativo)
  - unPunto (objeto Punto)

2. La clase Punto ha sido codificada previamente con dos atributos (x e y) y sus métodos constructores, getters y setters.

- Se imprimirá el contenido de la sesión antes de rellenarse.

- Este script generará un enlace a RecuperaSesion.java en el que se recuperará la sesión.

2.) RecuperaSesion.java

- En este script se recupera la sesión creada en el primer script y se visualiza el contenido de las 6 variables contenidas en su interior.

Nota: EL código de la clase Punto se incluye con el enunciado, además se incluye un esqueleto del programa, debiendo rellenar los huecos con el código correspondiente. En su hoja de respuesta incluirá el número del comentario donde incluiría el código, y el desarrollo del mismo.

CLASE PUNTO:

```
package auxiliar;

public class Punto {
   private int x;
   private int y;

   public Punto() {
   }

   public Punto(int x, int y) {
      this.x = x;
      this.y = y;
   }

   public void setX (int x) {
      this.x = x;
   }

   public int getX () {
      return this.x;
   }

   public void setY (int y) {
      this.y = y;
   }

   public int getY () {
      return this.y;
   }
}
```

Script CreaSesion.java

```
package sevlets;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import java.util.Enumeration;
import java.util.LinkedHashMap;
import java.util.Map;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
import auxiliar.Punto;

@WebServlet("/CreaSesion")
public class CreaSesion extends HttpServlet {

   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      try {
         // 1.1) Se vacía la sesión si existe

         // 1.2) Se crea una sesión asociada a la petición

         // 1.3) Se imprime el contenido de la sesión vacía

         // 1.4) Se registran variables de sesión (objetos de diferentes clases)

         // 1.5) Se crea y  visualiza el enlace al script RecuperaSesión

      }
      catch (Exception e){
         out.println("Se produce una excepción <br />");
         out.println(e.getMessage());
      }
   }
  @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
   }
}
```

Script RecuperaSesion.java

```
package sevlets;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet(name = "RecuperaSesion", urlPatterns = {"/RecuperaSesion"})
public class RecuperaSesion extends HttpServlet {
```

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {

            // 2.1) Se recuperan las variables de sesión previamente creadas

           // 2.2) Se imprime el contenido de la sesión

        catch (Exception e){
            out.println("Se produce una excepción <br />");
            out.println(e.getMessage());
        }
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    }
}
```

Ejercicio 4 (2,5 ptos)

Dadas las siguientes tablas de una base de datos:

VEHICULO

MATRICULA	MARCA	MODELO	PERSONA
2345AAA	Seat	Altea	NULL
1234BBB	Opel	Astra	15
5555CCC	Seat	Ibiza	NULL
9876DDD	Lexus	LC	10
1111FFF	Seat	Ibiza	NULL
2222GGG	Opel	Astra	20

EMPLEADO

ID	NOMBRE	DNI	Jefe
10	ANA MARTIN	1111A	NULL
15	JUAN LOPEZ	2222B	10
20	CARMEN DIAZ	3333C	10
25	ERNESTO GOMEZ	4444D	20
30	SILVIA GONZALEZ	5555E	15
35	FERNANDO SIERRA	7777F	15

ASIGNACION

ID	CODIGO
10	101
15	101
20	101
10	102
15	102
30	102
35	102
20	103
25	103
40	103

PROYECTO

CODIGO	Nombre
101	GRANATE
102	RUBI
103	ESMERALDA

1. Escribe las sentencias del Lenguaje de Definición de Datos necesarias para crear las tablas anteriores. Con las siguientes especificaciones:
    a. Cuando un proyecto es eliminado o modificado, se eliminan o modifican automáticamente las asignaciones correspondientes.
    b. Si un empleado se elimina o actualiza, se eliminan o actualizan las asignaciones de proyecto que pudiera tener.
    c. Cuando un empleado es eliminado, el vehículo que pudiera tener concertado no es eliminado y si la persona actualiza su identificación, automáticamente se actualiza también en su vehículo si lo tuviera.
    d. Cuando un empleado se elimina, su jefe sigue permaneciendo. Y si la persona actualiza su identificación, se actualiza automáticamente para todos aquellos que esa persona sea su jefe.
2. Realiza las siguientes consultas con el lenguaje SQL. Una sola sentencia SQL por cada apartado:
    2.1. Listado de los empleados que tienen jefe y nombre de su jefe.
    2.2. Listado de todas las personas y si tienen vehículo asignado matrículas del mismo.
    2.3. Nombre y código de proyecto del proyecto que tenga más personal asignado.