Ejercicio 1

1. Sea una Tabla Hash como la estudiada durante el curso con las siguientes características:

- Capacidad = 5
- Gestión de colisiones mediante direccionamiento cerrado con prueba cuadrática. c1=0, c2=1
- Función hash, el Módulo
- Los elementos de información son pares <K,V>, donde K es un valor numérico que representa la clave (es decir, la operación *hashValue* aplicada sobre elementos de ese tipo devolverá dicho valor), y V es la información asociada.

En un momento determinado la tabla se encuentra en la siguiente situación (donde el estado 'V' indica que la posición de la tabla está vacía, 'O' que está ocupada, y la cadena '----' indica que no existe elemento de la tabla almacenado en esa posición.):

	Elementos	Estados
0	-----	V
1	[321,Juan]	O
2	-----	V
3	-----	V
4	-----	V

Se pide representar el estado de la tabla después de las siguientes operaciones:

1. Insertar([582,Pepe])
2. Insertar([332,Ana])
3. Borrar(582)
4. Insertar([761,Pedro])
5. Insertar([146,Eva])
6. Insertar([761,María])

0->

1->

2->

3->

4->

2. Dado un mapa de tamaño 20 donde las colisiones se resuelven con prueba lineal y la función de dispersion es h(k) = k mod TAM_TABLA, siendo k la clave, represente el estado de dicho mapa después de insertar las siguientes claves: 4381, 1323, 6183, 4119, 4319, 9619, 1919. Describa si existe algun problema que puede reducir la eficiencia de esta tabla y, en caso afirmativo, cómo se podría minimizar.

3. El árbol rojo-negro resultante de insertar sucesivamente la siguiente secuencia de claves [8, 3, 6, 21, 15, 17, 16, 44] da como resultado el siguiente preorden:

a. 15, 6, 3, 8, 17, 16, 21, 44
b. 15, 6, 3, 8, 21, 17, 16, 44
c. 15, 6, 3, 8, 21, 16, 17, 44
d. 15, 6, 3, 8, 17, 16, 44, 21

Ejercicio 2.

Con la llegada de Netflix a España, es necesario disponer de un gestor para los contenidos disponibles. Netflix se caracteriza por almacenar dos tipos de contenidos: películas y series. En este gestor nos vamos a centrar en el contenido de tipo película. De cada película se almacena: título, año y tipo (acción, ciencia ficción, comedia, documental, drama, española, europea, familiar, premiada, romántica, terror y/o thriller), teniendo en cuenta que una película puede ser de más de un tipo.

Se desea implementar un gestor de Netflix que sea capaz de almacenar y gestionar los datos de películas y series en estructuras de datos de manera que la consulta del catálogo sea eficiente. Las funcionalidad debe ser la siguiente:

- Añadir contenido: dado un nombre de una película, un año de publicación, y sus géneros, se almacenará en el gestor para su posterior manejo de forma eficiente.
  ```
  public void addContent(String title, int year, ArrayList<String> types);
  ```
- Consulta por nombre: dado un nombre de una película, se recuperarán todas las que coincidan con el título dado.
  ```
  public Iterable<Movie> findTitle(String title)
  ```
- Consulta por año: dado un año, se devolverán todas las películas almacenadas que se encuentren en dicho año.
  ```
  public Iterable<Movie> findYear(int year)
  ```
- Consulta por tipo/s: dado un tipo o conjunto de tipos, se devolverán todas las películas que pertenezcan al/a los tipo/s dado/s.

  ```
  public Iterable<Movie> findType(String type)

  public Iterable<Movie> findType(Set<String> type)
  ```

Ejercicio 3

La clase `SumBinaryLevels` tiene la capacidad de sumar los valores de todos los nodos de unos niveles determinados de un árbol *n*-ario de enteros. Los niveles que se desean sumar se pasarán como argumento al método `sumLevels()` en forma de array de enteros, siendo el primer nivel (raíz) el nivel 0. El resultado debe ser la suma de todos los nodos situados en los niveles contenidos en el array. Para implementar el método `sumLevels()` se permite añadir cualquier atributo, estructura de datos o método a la clase `SumBinaryLevels`. Además, no se valorará ni penalizará el uso de recursividad y se podrá utilizar cualquier estructura de datos, método auxiliar o atributo que se considere necesario

Ejercicio 4

Una de las tareas más importantes en el análisis de redes sociales es la identificación de personas importantes. Una red social se compone de una colección de usuarios donde algunos de ellos están conectados y otros no. Para saber cuáles de estos usuarios son importantes en un grado k, siendo k un número entero, se sigue el siguiente procedimiento.

1. Se eliminan todos los usuarios de la red que tengan un número de conexiones menor que k. Es importante tener en cuenta que la eliminación de un usuario puede hacer que el número de conexiones de otro usuario pase a ser también menor que k, por lo que el procedimiento debe repetirse hasta que no quede en el grado ningún usuario con un número de conexiones menor que k.
2. Una vez eliminados esos usuarios, es necesario identificar en cuántos grupos se ha partido la red: si se mantiene formando un solo grupo, entonces los usuarios eliminados no eran importantes. Si por el contrario, la red queda formada por más de un grupo, los usuarios eliminados eran importantes en grado k. Un grupo se puede definir como un conjunto de usuarios que están conectados entre sí pero no con usuarios de otros grupos.

Para realizar este análisis, se proporciona la clase `SocialNetwork`, la cual debe ser implementada por el alumno. En particular, se pide:

a) **[1 punto]** Definir la estructura de datos necesaria para almacenar la red social e implementar los métodos `addUser` y `makeFriends`.

b) **[1 punto]** Implementar el método `filter`, el cuál eliminará a todos los usuarios de la red con un número de conexiones menor que k, siendo k un parámetro del método. Este método devolverá los usuarios eliminados.

c) **[2 puntos]** Implementar el método `groups`, el cuál indicará en cuántos grupos se encuentra dividida la red.