Práctica Individual 1 – Ejercicios iterativos

1.
```
public static List<Integer> ejemploI (List<List<Integer>> listas) {
        return listas
                .stream()
                .flatMap(lista -> lista.stream())
                .filter(e -> (Math2.esPrimo(e)))
                .collect(Collectors.toList());
}
```

2.
```
public static String ejemploN(Integer limit){
    return Stream.iterate(1,x->x<=limit,x->Math2.siguientePrimo(x))
                .map(x->x*x)
                .map(x->x.toString())
                .collect(Collectors.joining("\n"));
}
```

3. Un punto es un tipo con las siguientes propiedades:

- X, Double, básica, individual
- Y, Double, básica, individual
- Cuadrante, Cuadrante, derivada, individual. Cuadrante se define como un enumerado que puede tomar los valores: PRIMER_CUADRANTE, SEGUNDO_CUADRANTE, TERCER_CUADRANTE, CUARTO_CUADRANTE.

```
public static Map<Punto2D.Cuadrante,Double> ejemploU(List<Punto2D> l){
    return l.stream()
            .collect(Collectors.groupingBy(Punto2D::getCuadrante,
            Collectors.<Punto2D,Double>reducing(0.,x->x.getX(),(x,y)->x+y)));
}
```

Tenga en cuenta que:

- Para cada ejercicio debe leer los datos de entrada de un fichero, y mostrar la salida por pantalla. Dicha lectura debe ser independiente del algoritmo concreto que resuelva el ejercicio.
- La solución tiene que ser acorde al material de la asignatura proporcionado.

SE PIDE resolver de forma <u>eficiente</u>:

- Analice el código que se muestra y proporcione una solución iterativa equivalente (usando while) tanto en C como en Java.

Cada una de las entregas debe incluir:

- Proyecto en eclipse con las soluciones en C.
- Proyecto en eclipse con las soluciones en Java.
- Memoria de la práctica, que debe contener:
    - Código realizado

- Volcado de pantalla con los resultados obtenidos para las pruebas realizadas, incluyendo al menos los resultados obtenidos para los tests proporcionados.