Ejercicio 1

Utilizamos la siguiente función para buscar un real en un array de números reales:

```
1       public static int binSearch (double[] list, double item) {
2
3         int bottom = 0;
4         int top = list.length - 1;
5         int middle;
6         boolean found = false;
7         int location = -1;
8
9         while (bottom <= top && !found) {
10
11          middle = (bottom + top) / 2;
12
13          if(list[middle] == item) {
14            found = true;
15            location = middle;
16          } else if (list[middle] < item) {
17            bottom = middle + 1;
18          } else {
19            top = middle - 1;
20          }
21        }
22        return location;
23      }
```

Si el array que se pasa como parámetro es:
{1.0,3.0,7.0,8.0,11.0,16.0,21.0,25.0} cuantas veces se ejecuta el cuerpo del *while* si el número buscado es 3.0.

While se ejecuta 2 veces

¿Si el orden del array *list*, fuese descendente como deberíamos cambiar el método?

Hay que cambiar la línea 16 el > por <.

Ejercicio 2

Tenemos las siguientes clases:

```
class Nodo {
   protected int clave;
   protected int valor;
   protected Nodo iz,der;
```

```
…
}
class BST {
  private Nodo raiz;
…
}
```

Como podemos implementar un algoritmo recursivo que nos devuelva la máxima profundidad del árbol binario.

En la clase Nodo tenemos

```
int profundidad() {
  if (iz == null && der == null)
    return 1;
  int profIz=0;
  if (iz i= null) profIz=iz.profundidad();
  int profDer=0;
  if (der i= null) profDer=der.profundidad();
  if (profIz > profDer)
    return profIz+1;
  else
    return profDer+1;
}
```

En la clase BST tenemos:

```
int profundidad() {
  return raiz.profundidad();
}
```

Ejercicio 3

Supón que una clase implementa la siguiente función *hashCode*. Tendremos errores de compilación?, y de ejecución?, es esta una buena implementación de hashCode?

```
public int hashCode() {
   return 17;
}
```

No hay errores de compilación ni de ejecución, pero la implementación es inútil para utilizar en un hash map, porque todas las claves colisionan en una misma entrada del array.

Ejercicio 4

Empelando la clase *HashMap<String,Integer>()* de la biblioteca de java (una implementación de tablas hash con un string de clave y un entero de valor)

como podemos implementar un agenda de teléfonos. Consultar esa clase en la documentación de java (http://docs.oracle.com/javase/6/docs/api/). Como podemos crear una función para insertar un nueva entrada en la agenda, tomando como entradas el nombre de la persona y su teléfono. Como podemos obtener un número de teléfono a partir de un nombre.

```
class Agenda {
  private HashMap<String,Integer> miAgenda=
                                          new HashMap<String,Integer>();

  public void nuevoContacto(String nombre, int telefono)
                    throws ErrorAgenda {
    if (miAgenda.containsKey(nombre))
       throw new ErrorAgenda("Usuario ya dado de alta");
    miAgenda.put(nombre,teléfono);
  }

  public int obtenerTelefono(String nombre) throws ErrorAgenda {
    if (imiAgenda.containsKey(nombre))
           throw new ErrorAgenda("Usuario no dado de alta");
    return miAgenda.get(nombre);
  }
}
```