Ejercicio 1

Dada la siguiente definición recursiva:

$$suma2(a,i,j) = \begin{cases} a[i] & si\ \ j-i=1 \\ 2*suma2(a,i,m)-suma2(a,i,m)+a[m] & e.o.c \end{cases}$$

$$m = \frac{i+j}{2}$$

Donde a será un vector de enteros. Se pide:

a) Implementar en Java una función recursiva no final.
b) Optimice la función para que realice una sola llamada recursiva.
c) Implementar una función recursiva final en Java.
d) Implementar una función iterativa en Java.
e) Defina los tamaños, caso mejor y peor, y calcule los T(n) para las funciones implementadas en los apartados `a)`, c) y `d)`.

NOTA: Para todos los apartados la llamada inicial será `suma2(a,0,a.length*2-1)`.

SOLUCIÓN

a)

```
public static int suma2(int a[]) {
        return suma2(a, 0, a.length*2-1);
}

public static int suma2(int a[], int i, int j) {
        int aux = -1;
        if (j-i==1) {
                aux = a[i];
        } else {
                int m = (i+j) /2;
                aux = 2*suma2(a,i,m) - suma2(a,i,m)+ a[m];
        }
        return aux;
}
```

b) Transformación equivalente utilizando solo una llamada recursiva:

```
public static int sumamOpt(int a[]) {
         return sumamOpt(a, 0, a.length*2-1);
}

public static int sumamOpt(int a[], int i, int j) {
        int aux = -1;
        if (j-i==1) {
                aux = a[i];
        } else {
                int m = (i+j) /2;
                aux = sumamOpt(a,i,m) + a[m];
        }
        return aux;
}
```

c) Transformación final:

```
public static int sumamFinal(int a[]) {
         return sumamFinal(a, 0, a.length*2-1,0);
}

public static int sumamFinal(int a[], int i, int j, int acum) {
        int aux = -1;
        if (j-i==1) {
                aux = acum + a[i];
        } else {
                int m = (i+j) /2;
                aux = sumamFinal(a,i,m, acum + a[m]);
        }
        return aux;
}
```

d) Transformación iterativa:

```
public static int sumamIter(int a[]) {
         int i = 0;
         int j =  a.length*2-1;
         int acum =0;
         while(!(j-i==1)) {
                int m = (i+j) /2;
                acum = acum + a[m];
                j = m;
        }
        return acum + a[i];
 }
```

e)

El tamaño será:

$$n = j - i$$

Para el apartado a):

$$T(n) = 2T(n/2) + O(1) \Rightarrow T(n) = O(n)$$

Para el apartado b):

$$T(n) = T(n/2) + O(1) \Rightarrow T(n) = O(\lg n)$$

Para el apartado c):

$$T(n) = \sum_{x=1*2^i}^{n} k1\left(x^k (\log n)^p\right) + k2 = [k = 0, p = 0] \cong (\ln n)^{p+1}$$

$$T(n) = o(\lg n)$$