PROBLEMA 1: Recursividad

Dada la siguiente función recursiva:

$$func(int\ m, int\ n) = \begin{cases} aux[m] & si\ m = 0 \\ func(m-1, aux[m]) & si\ m > 0\ y\ n = 0 \\ func(m-1, aux[m]) + aux[m] & si\ m > 0\ y\ n > 0 \end{cases}$$

Siendo *aux* un array de enteros cuyo tamaño es mayor que *m* y *n*.

SE PIDE

1. Implementar en Java la función recursiva *func*.
2. Definir la función recursiva final.
3. Implementar en Java la función recursiva final.
4. Implementar en Java la función iterativa.
5. Defina los tamaños, caso mejor y peor, y calcule los T(n) para las funciones implementadas en los apartados 1), 3) y 4).

Solución:

1. Implementar en Java una función recursiva no final.

```
public static Integer func(Integer m, Integer n) {
            Integer res = 0;
            if (m == 0) {
                  res = aux[m];
            } else if (m > 0 && n == 0) {
                  res = func(m - 1, aux[m]);
            } else {
                  res = func(m - 1, aux[m]) + aux[m];
            }
            return res;
      }
```

2. Defina la función recursiva final.

funcF(int m, int n, int acu)

$$= \begin{cases} acu + aux[m] & si\ m = 0 \\ funcF(m-1, aux[m], acu) & si\ m > 0\ y\ n = 0 \\ funcF(m-1, aux[m],\ acu + aux[m]) & si\ m > 0\ y\ n > 0 \end{cases}$$

3. Implementar en Java la función recursiva final.

```
public static Integer funcF(Integer m, Integer n) {
            return funcfinal(m, n, 0);
      }
public static Integer funcfinal(Integer m, Integer n, Integer acu)
{
            Integer res = 0;
            if (m == 0) {
                  res = acu + aux[m];
            } else if (m > 0 && n == 0) {
                  res = funcfinal(m - 1, aux[m], acu);
            } else {
                  res = funcfinal(m - 1, aux[m], acu + aux[m]);
            }
            return res;
      }
```

4. Implementar la función iterativa en Java.

```
public static Integer funciterativa(Integer m, Integer n) {
            Integer res = 0;

            while (m > 0) {
                  if (n > 0) {
                        res = res + aux[m];
                  }
                  n = aux[m];
                  m--;
            }

            return res + aux[m];

      }
```

5. Defina los tamaños, caso mejor y peor, y calcule los T(n) para las funciones implementadas en los apartados a), c) y d).

Tamaño: m

No hay caso mejor ni peor ya que la recursividad depende del tamaño m.

$T(n) = T(n-1) + k = O(n)$