Estructuras de Datos
Tarea de creatividad 3: Transformacion de la recursión

Bonilla Ruiz Roberto Adrián
Num. Cta. 31721903-8

Fecha de entrega: Lunes 26 de Octubre

1. Escribe en código Java un método recursivo de cola con el nombre ***recursiveMultiplyChar*** que reciba una variable de tipo char y un entero no negativo n y regrese una cadena que consta de n repeticiones continuas de c.

```
public static String recursiveMultiplyChar (char c, int n){
        String cadena ="";
        if(n<=0){
            return cadena;
        }
        System.out.print(cadena+=c);
        return recursiveMultiplyChar(c,n-1);
}
```

---

2. Escribe en código Java otro método iterativo ***iterativeMultiplyChar*** que haga exactamente lo mismo que el recursivo de cola.

```
public static String iterativeMultiplyChar (char c, int n){
        String cadena="";
        while(n>0){
            cadena+=c;
            n--;
        }
        System.out.print(cadena);
        return cadena;
}
```

---

* *Punto extra.* Analiza la complejidad del algoritmo iterativo visto en clase que invierte los elementos de un arreglo.

```
1. private static void invierteRec(double[]arreglo, int primero, int ultimo){
       if(primero < ultimo){                                    (1)
           double temp = arreglo[primero];.                     (2)
           arreglo[primero] = arreglo[ultimo];                  (3)
           arreglo[ultimo] = temp;                              (4)
           invierteRec(arreglo,primero+1,ultimo-1);             (5)
       }
   }
```

(1) 2 operaciones: Declaracion y asignacion
(2) 2 operaciones: Acceso y asignación
(3) 2 operaciones: Acceso y asignación
(4) 1 operacion: Acceso a método El tiempo de ejecución total es f(n) = 2 + 6n

2.
```
private static void invierteIterativo(double[] arreglo){
    int primero = 0;                          (1)
    int ultimo = arreglo.length-1;            (2)
    while(primero < ultimo){                  (3)
        double temp = arreglo[primero];       (4)
        arreglo[primero] = arreglo[ultimo];   (5)
        arreglo[ultimo] = temp;               (6)
        primero++;                            (7)
        ultimo--;                             (8)
    }
}
```

(1) 2 operaciones: Declaracion y asignacion
(2) 3 operaciones: Declaración, acceso y asignación
(3) n iteraciones: primero <ultimo
(4) 3 operaciones: Declaración, acceso y asignación
(5) 1 operación: Asignación
(6) 1 operación: Asignación
(7) 2 operaciones: Acceso y suma
(8) 2 operaciones: Acceso y suma
$\therefore$ El tiempo de ejecución total es f(n) = 2 + 5n