___ METODOLOGÍA DE LA PROGRAMACIÓN (2º cuatrimestre). Curso 02-03
___ PROGRAMACION 1 (Plan viejo)

NOMBRE:_____APELLIDOS:_____

TIEMPO 2:00 h

Los alumnos de Metodología de la Programación deben contestar las preguntas [1, 2, 3, 4, 5, 6]
Los alumnos de Programación 1 deben contestar las preguntas [1, 2, 3, 4, 5]
Los alumnos de Laboratorio de programación deben contestar la pregunta [7]

1) Resuelve la siguiente ecuación de recurrencia. (3 ptos)

$$T(n) = \begin{cases} 1 & n \leq 2 \\ 9 \cdot T\left(\frac{n}{4}\right) & n > 2 \end{cases}$$

Resultado:

T(n) =

Nota.- Los cálculos hay que presentarlos en hoja a parte.

2) Considera los siguientes algoritmos (uno recursivo y otro iterativo) y responde a las preguntas que se indican a continuación (3 ptos)

```
Algoritmo Exa1 (x,y:R):R
METODO:
    SI y = 0 ENTONCES Exa1 ← 0
    SINO
        SI y mod 2 ≠ 0 ENTONCES  // impar
            Exa1 ← Exa1(2x,⌊y/2⌋) + x
        SINO
            Exa1 ← Exa1(2x,⌊y/2⌋)
        FSI
    FSI
FExa1
```

```
Algoritmo Exa2 (x,y:R):R
METODO:
    Res ← 0
    MIENTRAS y > 0 HACER
        SI y mod 2 ≠ 0 ENTONCES  // impar
            Res ← Res + x
        FSI
        x ← 2x
        y ← ⌊y/2⌋
    FMIENTRAS
    Exa2 ← Res
FExa2
```

a. ¿Implementan los dos algoritmos la misma función?

b. Suponiendo que el valor y es de la forma $y = 2^k$. ¿Qué algoritmo tiene un mayor coste temporal en asignaciones sobre cualquier elemento? En caso de que sean diferentes ¿Qué relación existe entre los costes de cada algoritmo?

c. Y si se consideran únicamente el coste en comparaciones sobre cualquier elemento (siendo $y = 2^k$) ¿Qué algoritmo tiene un mayor coste temporal? En caso de que sean diferentes ¿Qué relación existe entre los costes de cada algoritmo?

d. ¿Cuál es el coste espacial de cada uno de los algoritmos?

e. Si en vez de considerar que $y = 2^k$ resulta que $y = 2^k-1$ ¿Variará el coste temporal en asignaciones de alguno de los dos algoritmos? En caso afirmativo indica que algoritmo y como variará su coste.

f. Considerando la misma situación anterior (que $y = 2^k-1$) ¿Variará el coste temporal en comparaciones de alguno de los dos algoritmos? En caso afirmativo indica que algoritmo y como variará su coste.

3) Responde a las siguientes preguntas sobre algoritmos de ordenación. (3 ptos)

Considera los tres algoritmos de ordenación directa (Inserción, Selección e Intercambio) y un vector desordenado de n elementos. Si detenemos el proceso de ordenación después de situar un elemento 'i' cualquiera.

a. ¿Cuál de los tres algoritmos NO presenta un subvector [1..i] ordenado (considerando únicamente los elementos del 1 al i)?

b. ¿En cuál de los tres algoritmos el subvecotor [1..i] está correctamente ordenado respecto al vector total [1..n]? Es decir, ¿en cuál los valores del 1 al i se encuentran en su posición final?

c. ¿Cuál de los tres algoritmos se basa en la búsqueda del mínimo?

Considera ahora los tres algoritmos de ordenación rápida (Mergesort, Quicksort, Heapsort).

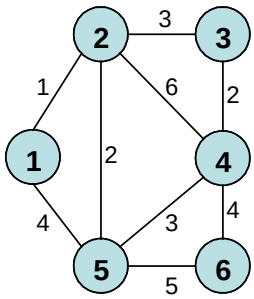
d. ¿Cuál de ellos va generando un vector correctamente ordenado?

e. ¿Alguno de estos algoritmos de ordenación no se basa en un planteamiento recursivo? ¿Cuál?

NOMBRE:_____APELLIDOS:_____

f. Indica el tipo de esquema recursivo en el que se basan

4) Escribe una versión del algoritmo de Kruskal para grafos no dirigidos con pesos que utilice una representación del grafo basada en una matriz de adyacencia, de forma similar a la versión implementada en Prim. Recuerda que la matriz de adyacencia representará el peso del arco que une dos nodos o -1 si no existe arco (ver figura) (4 ptos).



	1	2	3	4	5	6
1		1	-1	-1	4	-1
2	1		3	6	2	-1
3	-1	3		2	-1	-1
4	-1	6	2		3	4
5	4	2	-1	3		5
6	-1	-1	-1	4	5	

En la implementación NO puedes utilizar ningún vector auxiliar aunque si variables (Pista: utiliza la diagonal principal para almacenar el número de componente conexa a la que pertenece el nodo). El algoritmo NO puede contener llamadas a funciones externas, todos los procesos deben estar incluidos.

Utiliza la siguiente cabecera para el algoritmo (n representa el número de nodos):

```
Algoritmo kruscal // Con matrices de adyacencia
DATOS: G[1..n,1..n] de R          // Matriz de adyacencia del grafo.
       T[1..n-1] de record:       // Vector que almacena los arcos que constituyen el
           Nodo1: N               // camino mínimo. El número de arcos del camino
           Nodo2: N               // mínimo es n-1
METODO:
```

Escribe la solución en un folio aparte.

5) Diseñar un algoritmo de BACKTRACKING que genere todas las palabras "*palíndromas*" de n letras (que se leen igual de izquierda a derecha que de derecha a izquierda) que se puedan formar con los caracteres: ['a', 'b', 'e']. Recuerda que, aunque el problema se puede resolver de diferentes formas, hay que implementar una solución basada en backtracking (4 ptos).

Se pide:

a. ¿Qué tipo de backtracking se trata?

b. ¿De que orden será el coste temporal en el peor caso? (No es necesario calcular el coste exacto)

c. ¿De que orden será el coste espacial?

d. Si en vez de tener 3 letras tenemos 4 letras ¿Se modificaría alguno de los costes?. ¿Cuál o cuales?. Indica el nuevo valor del coste.

e. Implementa el algoritmo en un folio a parte:

6) Dada la siguiente función iterativa: (3 ptos)

```
func prod(a: entero; v: vector[1..N] de enteros) dev p: vector[1..N] de enteros
{Pre:  cierto }
i := N;
{Inv: ???; Dec: ???}
[ i ≠ 0 → p[i] := a * p[i];
    i := i – 1;
]
{Post: p = ∀ α:1 ≤ α ≤ N: p[α] = a * p[α] }
```

Se pide:

a. Escribe la función de cota y el invariante.

b. Verifica que el invariante se cumple al entrar en el bucle.

c. Verifica que la postcondición se cumple al acabar el bucle

NO hace falta verificar el resto del programa (Responde esta pregunta en un folio aparte).

7) En la práctica 2 se analizaba la búsqueda binaria y algunas variaciones de la misma. Responde a las siguientes preguntas (3 ptos)

a. Indica si las siguientes expresiones son correctas, o no, para determinar la posición del pivote en el algoritmo de búsqueda binaria. (Rodea la respuesta correcta).

$$p = Ini + \left(\frac{Fin - Ini}{2}\right) \quad \text{V ó F} \quad ; \quad p = Fin - \left(\frac{Fin - Ini}{2}\right) \quad \text{V ó F}$$

b. En el caso del algoritmo de búsqueda ternaria, ¿podríamos utilizar las siguientes expresiones para calcular los dos pivotes centrales?

$$p_1 = Ini + \left(\frac{Fin - Ini}{3}\right) \text{ y } p_2 = Ini + 2 \cdot \left(\frac{Fin - Ini}{3}\right) \quad \text{Si ó No}$$

c. ¿Que algoritmo, el de búsqueda binaria o el de búsqueda ternaria, es mejor en comparaciones?. ¿Son sus costes del mismo orden?. Indica de que orden es cada uno de ellos.