Resolver cada pregunta en una hoja distinta.
No hay que entregar esta hoja con el examen.

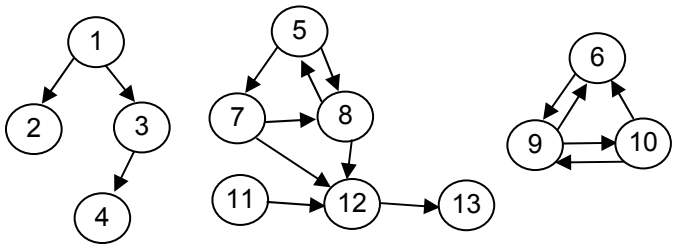
1. (2,5 puntos) Considera la estructura de relaciones de equivalencia vista en clase, mediante árboles de punteros al padre, usando las técnicas de balanceo de árboles y compresión de caminos. Mostrar un ejemplo, con sucesivas operaciones de unión sobre una relación inicialmente vacía, donde uno de los árboles crezca hasta profundidad 3. Se deben mostrar en cada paso los valores de la tabla y la estructura de árboles asociada.

2. (2,75 puntos) Un árbol B de orden **p** = **n**+1 se usa para almacenar valores enteros. La definición del tipo de datos es la siguiente:

```
tipo
    nodo= registro
            valores: array [1..n] de entero
            punteros: array [0..n] de Puntero[nodo]
    finregistro
    ArbolB= Puntero[nodo]
```

   Si una entrada del nodo está vacía, el valor correspondiente vale -1. Para los nodos hoja, todos los punteros tienen valor NULO. Escribir un procedimiento para recorrer de forma ordenada (de menor a mayor) todos los elementos de un árbol B que estén entre dos valores dados **v1** y **v2** (con **v1**<**v2**), ambos inclusive, usando los tipos definidos. La cabecera del procedimiento será:
   operación Recorrido (raíz: ArbolB; v1, v2: entero)

3. (2,75 puntos) Tenemos un conjunto de páginas web que queremos poner en una página de enlaces. Pero nos damos cuenta de que algunas de estas páginas ya enlazan a otras del conjunto, de forma que podemos eliminar las ya enlazadas. Por ejemplo, supongamos que la página 1 enlaza la 2 y la 3; y que la 3 enlaza a la 4. Si seleccionamos la 1, no necesitamos poner 2, 3 ni 4. El objetivo es seleccionar el número mínimo de páginas tal que todas las páginas del conjunto están enlazadas directa o indirectamente. Escribir un algoritmo para resolver el problema. Se pueden suponer los algoritmos vistos en clase, siempre que quede claro su uso (cuáles son los parámetros de entrada, el resultado y el efecto del algoritmo).
   Datos del problema. El conjunto inicial tiene **n** páginas, enumeradas como {1, 2, ..., n}. La matriz de booleanos **M**, indica en cada posición **M[a, b]** si la página **a** enlaza a la **b**. La matriz **M** no es necesariamente simétrica.
   Ejemplo, con **n** = 13. Las páginas se representan con círculos y los enlaces mediante flechas. Una posible solución óptima (no la única) sería seleccionar: 1, 6, 7, 11.



4. (2 puntos) Suponer que tenemos las especificaciones formales de los TAD **Lista[T]** y **Natural**, con las siguientes operaciones:
   - **N=Natural**: *cero*, *sucesor*, *esCero*, *esMenor*, *suma*, *resta*, *multiplica*, *doble*, *mitad*.
   - **Lista[T]**: *vacía* (devuelve una lista sin elementos), *inserta* (dada una lista y un elemento, añade ese elemento en la primera posición de la lista).

Queremos añadir las siguientes operaciones al TAD **Lista[T]**: **sacaPares** (dada una lista cualquiera, devuelve otra lista en la que se ha eliminado el 2º elemento, el 4º, el 6º, etc., es decir, todos los elementos en posiciones pares), **primeraMitad** (dada una lista con *k* elementos, devuelve otra lista con los elementos 1º, 2º, ..., hasta el k/2-ésimo), **segundaMitad** (dada una lista con *k* elementos, devuelve otra lista con los elementos k/2+1-ésimo, k/2+2-ésimo, ..., hasta el k-ésimo). Escribir la sintaxis y la semántica de las operaciones añadidas. Se pueden incluir otras operaciones, pero será necesario especificarlas también. Nota: usar *mitad(k)* para obtener *k*/2.

Nota: La pregunta 4 no deben hacerla los alumnos que tengan aprobada la Práctica de Especificaciones Algebraicas con Maude. Ojo, la asignatura "Laboratorio de Programación" no convalida esta práctica.

Resolver cada pregunta en una hoja distinta.
No hay que entregar esta hoja con el examen.

1. (3 puntos) Considerar el siguiente algoritmo recursivo basado en divide y vencerás.

```
operación DV (n: entero): real
var m, k: entero
    d: real

        d:= 1.0 + n
        si n>1 entonces
                m:= n/2
                d:= d + DV(m)
                si m>1 entonces
                        k:= m/2
                        d:= d + 4*DV(k)
                        d:= d + 8*DV(k)
                finsi
        finsi
        devolver d
```

Se pide:

a) (1,2 puntos) Estimar el orden de complejidad del algoritmo, usando la notación asintótica que creas más conveniente.
b) (1 punto) Calcular el valor devuelto por el algoritmo mediante una fórmula explícita, es decir, no recursiva. Si aparecen más de 3 constantes, se puede dejar indicado el sistema de ecuaciones que habría que resolver.
c) (0,8 puntos) Estimar el orden exacto, Θ, del consumo de memoria del algoritmo.

2. (2 puntos) El Sudoku se ha convertido en el pasatiempo del verano. Se trata de una matriz de 9x9 celdas, donde en cada celda puede haber un número entre 1 y 9. Se añade la restricción de que cada número sólo puede aparecer una vez en cada fila, en cada columna y en cada bloque de celdas. Los bloques son grupos de 3x3 celdas adyacentes, como se muestran abajo (ver que, en total, hay 9 filas, 9 columnas y 9 bloques). Algunas celdas del Sudoku están inicializadas con ciertos valores y otras están vacías. El objetivo es completar las celdas vacías, con valores que cumplan las restricciones del juego.
Se pide diseñar un algoritmo voraz para intentar resolver el problema. Indicar cuáles son los candidatos, cómo es la función de selección, cómo es la inserción de un elemento en la solución, y las demás partes del esquema. Suponer que la entrada es una matriz de 9x9, donde un valor 0 indica una celda no inicializada.

	3		9	5	2	7		
6	5			8	1	2		
							5	
		8			6	4		7
	4	6	1		5	8	9	
7		3	2			5		
	8							
		9	8	2			1	5
		5	4	6	3		8	

Sudoku de ejemplo

http://websudoku.com

3. (2 puntos) Suponer que la descomposición recurrente (la fórmula recursiva de programación dinámica) para el problema de la mochila 0/1 se aplica usando una estrategia

descendente, es decir, un algoritmo de divide y vencerás (DV) que aplica directamente la recursividad. Escribir la fórmula y el algoritmo de DV que surge de la misma. Mostrar la ejecución del algoritmo de programación dinámica y el de DV sobre el siguiente ejemplo: **n**= 3; **M**= 8; **p**= {3, 4, 5}; **b**= {5, 6, 7}. En el caso del algoritmo de DV, se deberá mostrar el árbol de llamadas recursivas correspondiente. ¿Cuál es el resultado final en ambos? ¿Qué algoritmo consideras que sería más eficiente en este ejemplo concreto? ¿Y en general?

4. (3 puntos) Resolver el problema del Sudoku (ejercicio 2) mediante un algoritmo de backtracking o mediante ramificación y poda. Se deben utilizar obligatoriamente los esquemas algorítmicos vistos en clase. Explicar cómo es la representación de la solución, la forma del árbol, y las funciones genéricas del esquema correspondiente. Se pueden dar por supuestos los esquemas vistos en clase, sólo es necesario indicar aquellas partes que se consideren ambiguas.

Nota: La pregunta 2 no deben hacerla los alumnos que tengan convalidada la Práctica 4.