Nombre y apellido: LEANDRO ESTEBAN CATALDO

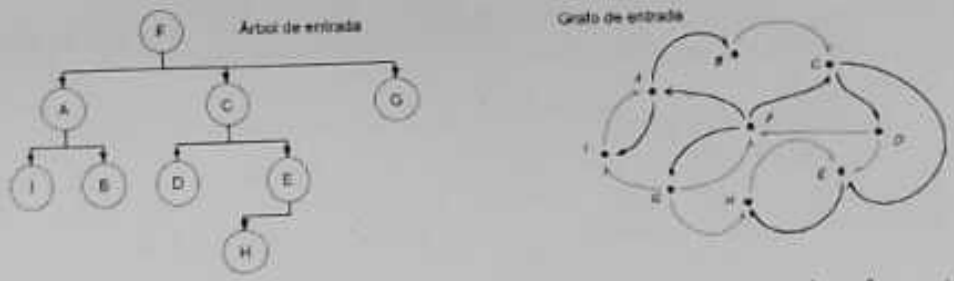
Ejercicios (puntaje)			Nota
1 (10/3 pts)	2 (10/3 pts)	3 (10/3 pts)	desaprobado

CONSIDERACIONES: RESOLVER CADA EJERCICIO EN UNA HOJA DIFERENTE. LOS EJERCICIOS QUE NO ESTÉN CORRECTOS EN UN 50% NO SUMARAN PUNTOS PARA LA NOTA FINAL. PARA CADA EJERCICIO DEBE **DEFINIR EL TIPO DE DATO** DE CADA TDA UTILIZADO. **EN TODOS LOS CASOS QUE UTILICE ESTRUCTURAS** QUE NO SEAN TDAs "ESTÁNDAR" **DEBE DEFINIR LOS STRUCTS**.

Ejercicio 1: Desarrollar una función y todas las necesarias para que dado un árbol binario y una clave de un nodo, **devuelva una pila (TDA Stack) con el camino desde la raíz a dicho nodo**. Ejemplo:

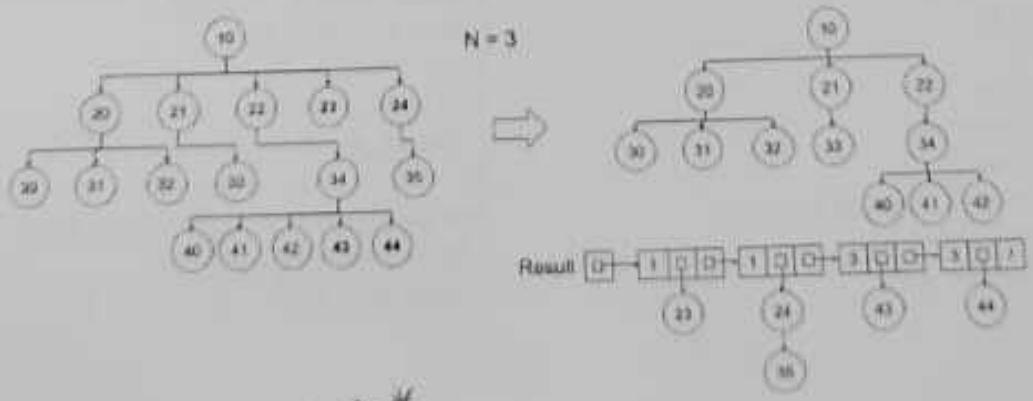
Encabezado función principal: `stack* binary_path (btn* root, int key);`

Ejercicio 2: Desarrollar una función y todas las necesarias para que dado un **árbol n-ario** (implementado con **TDA *Vector*** en cada nodo para acceder a los hijos) y un **grafo dirigido** (implementado con listas de adyacencias con **TDA *Vector*** y listas dinámicas enlazadas), **devuelva 1 si el árbol está incluido en el grafo, y 0 si no lo está.** NO TDA GRAPH
Ejemplo: Para el siguiente árbol y grafo la función devuelve 1.



Encabezado función principal: `int is_included (ntn* t, digraph* g, int cmp(t_elem, t_elem));`
NOTA: considerar que el elemento del árbol y del grafo es de tipo "**t_elem**" y existe una función de comparación.

Ejercicio 3: Desarrollar una función y todas las necesarias para que, dado un árbol n-ario (implementado con listas dinámicas enlazadas), se **reduzca el grado del árbol a un máximo de N (pasado por parámetro) moviendo los nodos sobrantes a una lista dinámica enlazada resultante**. Cada nodo de la lista debe tener el nivel del nodo removido y puntero al nodo, además la lista debe quedar ordenada por nivel. Ejemplo:



list *

Encabezado función principal: `list_node* ntn_reduce_degreee (ntn* root, int N);`