Para cada una de las siguientes situaciones, desarrolle un programa en `C` o en `Java` que la resuelva.

1. Árboles. En todos los casos se debe considerar opciones para poder escribir en la salida estándar el contenido del árbol en preorden, inorden y posorden.
   (a) Considere información que al menos contenga apellido paterno, apellido materno y nombre. Utilice árboles binarios para insertar la información considerando la comparación primero del apellido paterno, en caso de coincidir, comparación del apellido materno, en caso de coincidir, comparación del nombre y en caso de coincidir, actualizar el nodo correspondiente con el resto de la información. Se debe contar con una opción para poder buscar por apellido paterno, se debe mostrar toda la información del nodo encontrado. La comparación es una comparación de cadenas en orden lexicográfico.
   (b) Considere información que al menos contenga apellido paterno, apellido materno y nombre. Utilice árboles con seis ramas, vamos a considerar las dos primeras para visualizar el árbol como uno binario para el apellido paternos, las dos siguientes para visualizar el árbol como uno binario para el apellido materno y las dos últimas para visualizar el árbol como uno binario para el nombre. Para insertar la información, se le debe dar un tratamiendo en los diferentes pares de ramas de acuerdo al elemento a comparar: apellido paterno, apellido materno y nombre. En cada caso, antes de insertar la información, se debe comparar que no se repita el nombre completo. De repetirse, se debe actualizar la información restante. Se debe contar con una opción para poder buscar por apellido paterno o apellido materno o nombre y, de encontrarse el elemento, se debe mostrar toda la información del nodo encontrado. La comparación es una comparación de cadenas en orden lexicográfico.
   (c) Genere un árbol binario copleto. Para insertar información, se debe auxiliar de considerar en número de nodos que se tiene considerando el nodo dónde se encuentre como si fuera nodo raíz. Para agregar un nuevo nodo, se debe considerar el número que le correspondería, es decir, si ya hay diez elementos en un árbol, la nueva información a insertar le corresondería el nodo 11. Para saber en qué lugar se debe insertar, primero hay que ver el número de bits a considerar, es decir, considere:

```
nbits=0;
while(nesimonodo>0){
  nbits++;
  nesimonodo>>=1;
}
```

   para saber en que rama agregar el nodo, considere la siguiente observación:

```
aQueRama= nesimonodo & (nbits-1);
si nbits > 1 entonces
  si aQueRama es cero, revisar en la rama izquierda decrementando  nbits
  si aQueRama es uno, revisar en la rama derecha decrementando  nbits
si nbits es 1 entonces
  si aQueRama es cero, insertar en la rama izquierda decrementando  nbits
  si aQueRama es uno, insertar en la rama derecha decrementando  nbits
```

Escriba la información del árbol nivel por nivel.

(d) Dada una expresión en notación infija, como se describió en clase, poner la en un árbol y de ahí obtener un árbol con la derivada simbólica de la expresión dada. Se debe escribir en la salida estándar la derivada simbólica correspondiente a la expresión dada.

2. Grafos.

(a) Dado un grafo con o sin pesos y uno de sus vértices, indique los vértices adyacentes y, de ser el caso, los pesos de las aristas corresondientes.

(b) Dado un grafo, indique si se puede realizar un recorrido euleriano e indique los vértices que conforman dicho recorrido, de ser el caso.

(c) Dado un grafo, genere la matriz auxiliar que indique el reccorido de menor costo desde un nodo a otro indicando tal recorrido.

(d) Dado un grafo con pesos positivos y un vértice, obtenga los pesos mínimos del vértice dado a todos y cada uno de los demás vértices del grafo y sus recorridos. (se debe implementar lo visto en clase para este caso)