Función graphMatching (GrafoM, GrafoB)	/*GrafoM: Grafo muestra o grafo a buscar. GrafoB: Grafo donde se busca GrafoM. */
1. Generar códigos lv, le, lvev. codigosM <- generaCodigos(GrafoM) codigosB <- generaCodigos(GrafoB)	/*codigosM y codigosB son estructuras con las listas de códigos lv, le, lvev. */
2. Generar código canónico de GrafoM. FormaCanonica <- crearFormaCanonica(codigosM) AuxCanonica <- FormaCanonica	/*FormaCanonica es una lista con la forma canónica. AuxCanonica irá tomando la forma de las derivadas canónicas.*/

3. verificarPatrones(AuxCanonica, ListaPatrones)
//Se revisa si hay patrones en ListaPatrones que coinciden con
// AuxCanonica.
	/*Respetando orden de AuxCanonica y hacer previa verificación en ListaPatrones.
	Lista patrones contiene las estructuras encontradas.
	Si se encuentran múltiples coincidencias se crearán los patrones necesarios en ListaPatrones.*/

4. Buscar nuevos patrones con respecto a la nueva raíz. NuevasRaices(AuxCanonica, codigosB, ListaPatrones)	
5. Hacer matcheo de AuxCanonica con códigosB [GrafoB] Para cada patrón marcado se busca continuar el matching. $ListaPatrones_i$ <- Matching(AuxCanonica, codigosB, $ListaPatrones_i$)	
6. Verificar que todo arco en codigosB esté visitado; crear forma canónica derivada. Si visitaCompleta(codigosB)=FALSO entonces FormaDerivada <- formaCanonicaDerivada(AuxCanonica) AuxCanonica <- FormaDerivada Volver a paso 3. Fin_si	/*FormaDerivada es una permutación de la forma canónica.*/ //Todos o de acuerdo a especificaciones.
7. Reportar patrones encontrados.	
Fin_Función.	

Función generaCodigos (Grafo) 1. Generar lista de códigos lv de Grafo -> Codigos.lv 2. Generar lista de códigos le de Grafo -> Codigos.le 3. Generar lista de códigos lvev de Grafo -> Codigos.lvev 4. Generar lista de arcos del Grafo -> Codigos.arcos 5. Retornar Codigos. Fin_Función.	/*La salida es una estructura llamada Codigos, que contiene las listas de códigos lv, le, lvev y arcos de Grafo. lvev también contiene frecuencia de la etiqueta y un promedio de adyacencia. Codigos.arcos es una lista con los arcos y las propiedades de los arcos (grado de adyacencia, enumeración de vértices.)*/

Función crearFormaCanonica (codigosM)
 	
1. Ordenar la lista de arcos respecto a la mejor opción.
OrdenarconReglasPrioridad(codigosM.arcos, codigosM.lvev)
 	
2. Insertar el primer código de la lista codigosM.arcos en la lista Sucesores.
Sucesores <- $codigosM.arcos_1$
 	
3. Tomar el primer código en la lista Sucesores y marcar como visitado en codigosM.arcos.
 	
4. Añadir a la forma canónica el código que se marcó.
$FormaCanonica_i$ <- $codigosM.arcos_j$
 	
5. //Se elimina de la lista Sucesores el código usado y se insertan los //nuevos candidatos.
Marcamos en codigosM.arcos a los sucesores del primer código en la lista Sucesores.
Eliminamos el primer código en Sucesores //Ya se usó ese código.
Sucesores <- [Nuevos sucesores] //Se añaden los códigos recién
. // marcados como sucesores.
 	
6. Ordenar Sucesores.
OrdenarconReglasPrioridad(Sucesores, codigosM.lvev)
 	
7. Si todo arco en codigosM.arcos está marcado como visitado = FALSO, volver a paso 3.
 	
8. Retornar FormaCanonica.
 	
Fin_Función.
	/* Salida: lista FormaCanonica con la forma canónica del grafo GrafoM.*/
	
	/* FormaCanonica
	
	/* Un sucesor es un arco adyacente a otro arco ya visitado. En este caso se añaden los sucesores del último arco visitado, solo si no han sido marcados antes como sucesores. */
	
	/* La condición en paso 7 puede ser también, si Sucesores está vacía = FALSO, volver a 3. */

Función OrdenarconReglasPrioridad(ListaArcos, Listalvev)
 	
1. Se ordenan los arcos de mayor a menor prioridad respetando las siguientes reglas.
1.1. Grado de adyacencia de forma ascendente.
//Nos apoyamos de ListaArcos ya que contiene la adyacencia.
1.2. Frecuencia de aparición, descendente.
//Nos apoyamos de Listalvev ya que contiene las frecuencias.
1.3. Orden lexicográfico.
//Orden lexicográfico de los arcos.
1.4. Enumeración de vértices.
//ListaArcos contiene la información de enumeración de vértices.
 	
Fin_Función.
	/* Se ordenan los arcos del grafo para trabajar con ellos en un orden de prioridad que dictan las reglas:
	- Grado de adyacencia (ascendente).
	- Frecuencia de aparición (descendente).
	- Orden lexicográfico.
	- Enumeración.
	*/

Función verificarPatrones (FormaCanonica, ListaPatrones) 1. Si ListaPatrones cuenta con códigos de arcos que están en FormaCanonica, recorrer y marcar FormaCanonica hasta el primer código de arco que no se encuentre en ListaPatrones. Fin_Función.	/* Para todos los patrones posibles en ListaPatrones que comiencen de igual forma que FormaCanonica. */

Función NuevasRaices(AuxCanonica, codigosB, ListaPatrones) 1. Para cada arco en Codigos.arcos si Codigos.arcos = $AuxCanonica_1$, Crear nueva entrada en ListaPatrones. Marcar para ser expandidos junto con patrones verificados en el paso anterior. Fin_Función.	

Función Matching (AuxCanonica, codigosB, $ListaPatrones_i$) 1. Para cada patrón marcado en ListaPatrones, Colocarse en el último código marcado en $ListaPatrones_i$ e ir agregando a la lista Adyacentes, los arcos adyacentes a los códigos que ya se encuentran en ListaPatrones (sin repetir). 2. Colocarse en el último código de arco de AuxCanonica que está marcado en $ListaPatrones_i$, avanzar un código de arco más en AuxCanonica. 3. Si el arco actual de AuxCanonica está en Adyacentes, marcarlo en Adyacentes y CodigosB.arcos, añadir el arco a $ListaPatrones_i$,	

añadir sus arcos adyacentes a Adyacentes y eliminar el arco recién marcado en Adyacentes. 4. Regresar a 2. Fin_Función.	

Función formaCanonicaDerivada (AuxCanonica)
 	
1. Eliminar el primer código de la lista AuxCanonica.
//Para cambiar la raíz de la búsqueda.
 	
2. Recorrer y conservar los códigos que tuvieron coincidencias.
 	
3. De acuerdo a los códigos conservados, llenar lista de Sucesores correspondientes.
 	
4. Expandir a partir de los códigos conservados y la lista de Sucesores.
//Como en crearFormaCanonica. Puede ser una función sobrecargada.
 	
Fin_Función.
	/* Salida: devuelve una derivación o permutación de la forma canónica. */