HOJA N°

FECHA

```
public class BuscadorDeCaminos {
    private Grafo<String> bosque;
    public ListaGenerica<ListaGenerica<String>> recorridosMasSeguros(){
        ListaGenerica<ListaGenerica<String> recorridos = new ListaGenerica<ListaGenerica<String>>();
        if (!this.bosque.esVacio() && this.bosque != null){
            boolean [] visitados = new boolean visitados[this.bosque.listaDeVertices().tamanio()];
            boolean encontreCaperucita = false;
            ListaGenerica<String> camino = new ListaGenerica<String>();
            Vertice<String> v = new Vertice<String>();
            ListaGenerica<String> aux = new ListaGenerica<String>();
            aux.comenzar();
            while(! aux.fin() && !encontreCaperucita){
                v = aux.proximo();
                if(v.dato().equals("Caperucita")){
                    encontreCaperucita = true;
                }
            }
            if(encontreCaperucita){
                dfs(recorridos, visitados, camino, v.posicion());
            }
            return recorridos;
        }
```

```
private void dfs (ListaGenerica<ListaGenerica<String>> recorridos, boolean [] visitados,
ListaGenerica <String> camino, int i){
    visitados[i] = true;
    Vertice <String> v = this.bosque.listaDeVertices().elementoEn(i);
    camino.agregarFinal(v.dato());
    if( v.dato().equals("Casa Abuelita"){
        recorridos.agregarFinal(camino.clonar());
    }
    else {
        ListaGenerica<Vertice<String>> ady = this.bosque.listaDeAdyacentes(v);
        ady.comenzar();
        while(!ady.fin()){
            Arista<String> arista = ady.proximo();
            int j = arista.verticeDestino().posicion();
            if (!marca[j] && arista.peso() < 5){
                dfs(recorridos, visitados, camino, i);
                visitados[j] = false;
                camino.eliminarEn( camino.tamanio() );
            }
        }
    }
}
```

NOTA