Clase: GraphByMatrix		Método: initMatrix()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	la prueba comprueba que la matriz se inicializa correctamente.	El grafo sin su matriz de adyacencia.	matriz de adyacencia inicializada.

Clase: GraphByMatrix		Método: addVertex()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	la prueba comprueba que dos vertices, son agregados al grafo. en nuestro caso, vertices representados como ciudades: cali,bogota	La matriz inicializada sin sus vertices.	matriz con dos vertices agregados.

Clase: GraphByMatrix		Método: addEdge()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	la prueba comprueba que dos bordes, son agregado al grafo. en nuestro caso, bordes representados como carreteras: panamericana, troncal	La matriz inicializada con los vertices cali y bogota.	matriz con dos bordes entre sus vertices ya existentes.

Clase: GraphByMatrix		Método: RemoveEdge()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	la prueba comprueba que un borde, es eliminado correctamente del grafo. en nuestro caso, se eliminará el borde: panamericana	La matriz inicializada con los vértices cali y bogotá, y dos bordes agregados entre ellas: troncal y panamericana	matriz con un borde entre sus vértices ya existentes.

Clase: GraphByMatrix		Método: numEdgesOfVertex()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	la prueba comprueba si el numero de bordes entre dos vertices es correcto	La matriz inicializada con los vértices cali y bogotá, y dos bordes agregados entre ellas: troncal y panamericana	un entero que representa el valor correcto de bordes entre los vertice

Clase: GraphByMatrix		Método: isAdjacent()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	la prueba comprueba dos vertices tienen un borde entre ellas. “si dos ciudades tienen una carretera que las una”	La matriz inicializada con los vértices cali y bogotá, y dos bordes agregados entre ellas: troncal y panamericana	True. existen dos carreteras entre las ciudades cali y bogota.

Clase: GraphByLists		Método: addVertex()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	la prueba comprueba que dos vertices, son agregados al grafo. en nuestro caso, vertices representados como ciudades: cali,bogota	La lista inicializada sin sus vertices.	lista con dos vertices agregados.

Clase: GraphByLists		Método: addEdge()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	la prueba comprueba que dos bordes, son agregado al grafo. en nuestro caso, bordes representados como carreteras: panamericana, troncal	La lista inicializada con los vertices cali y bogota.	lista con dos bordes entre sus vertices ya existentes.

Clase: GraphByLists		Método: numEdgesOfVertex()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	la prueba comprueba si el numero de bordes entre dos vertices es correcto	La lista inicializada con los vértices cali y bogotá, y dos bordes agregados entre ellas: troncal y panamericana	un entero que representa el valor correcto de bordes entre los vertice

Clase: GraphByLists		Método: isAdjacent()	
Caso #	Descripción de la prueba	Estado Inicial	Resultado
1	la prueba comprueba dos vertices tienen un borde entre ellas. “si dos ciudades tienen una carretera que las una”	La lista inicializada con los vértices cali y bogotá, y dos bordes agregados entre ellas: troncal y panamericana	True. existen dos carreteras entre las ciudades cali y bogota.