Q1: Write a Gremlin command to create the given graph

Step 1: Creating a new graph and adding vertices

```
gremlin> graph=TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin> v1=graph.addVertex(id,1,label,"CS101")
==>v[1]
gremlin> v2=graph.addVertex(id,2,label,"CS201")
==>v[2]
gremlin> v3=graph.addVertex(id,3,label,"CS220")
==>v[3]
gremlin> v4=graph.addVertex(id,4,label,"CS420")
==>v[4]
gremlin> v5=graph.addVertex(id,5,label,"CS334")
==>v[5]
gremlin> v6=graph.addVertex(id,6,label,"CS681")
==>v[6]
gremlin> v7=graph.addVertex(id,7,label,"CS400")
==>v[7]
gremlin> v8=graph.addVertex(id,8,label,"CS526")
==>v[8]
```

Explanation: the open() method creates a new graph with 0 vertices and 0 edges. Vertices can be added with addVertex() method. ‘id’ and ‘label’ are reserved words and are required.

Step 2 : Adding edges to the graph

```
gremlin> v2.addEdge("prereq",v1,id,9)
==>e[9][2-prereq->1]
gremlin> v3.addEdge("prereq",v2,id,10)
==>e[10][3-prereq->2]
gremlin> v4.addEdge("prereq",v3,id,11)
==>e[11][4-prereq->3]
gremlin> v4.addEdge("coreq",v3,id,12)
==>e[12][4-coreq->3]
gremlin> v5.addEdge("prereq",v2,id,13)
==>e[13][5-prereq->2]
gremlin> v6.addEdge("prereq",v5,id,14)
==>e[14][6-prereq->5]
gremlin> v7.addEdge("prereq",v5,id,15)
==>e[15][7-prereq->5]
gremlin> v8.addEdge("prereq",v7,id,16)
```

```
==>e[16][8-prereq->7]
gremlin> v8.addEdge("coreq",v7,id,17)
==>e[17][8-coreq->7]
```

Explanation: Edges are added with addEdge() method. Providing a ‘label’ is mandatory followed by the vertex into which the edge goes and its unique ‘id’.

Step 3: Creating a graph traversal

```
gremlin> g=graph.traversal()
==>graphtraversalsource[tinkergraph[vertices:8 edges:9], standard]
```

Explanation: graph.traversal() method creates a traversal source which is used to traverse around the graph.

Q2: Write a query to output just the doubly-connected nodes

```
gremlin> g.V().as('a').out('coreq').as('b').select('a','b').by(label)
==>[a:CS420,b:CS220]
==>[a:CS526,b:CS400]
```

Explanation: This iterates over all the available vertices in a graph and picks out the vertices having an outgoing ‘coreq’ edge. It names the vertex which has an outgoing edge as ‘a’ and the vertex into which the edge goes as ‘b’. Finally by() outputs the vertices with their labels.

Q3: Write a query to output all the ancestors of a given vertex.

```
gremlin> g.V().has(label,'CS526').repeat(out('prereq')).until(outE().count().is(0)).path().by(label)
==>[CS526,CS400,CS334,CS201,CS101]
```

Explanation: This query starts from the vertex CS526 and repeatedly goes through all the vertices to which there is an ‘out’ edge with the label ‘prereq’ until it reaches one vertex which has 0 ‘out’ edges. It outputs the entire path with the help of labels.

Q4: Write a query that will output the max depth starting from a given node

```
gremlin> g.V().has(label,'CS101').repeat(__.in('prereq')).until(inE().count().is(0)).path().tail().unfold().count()
==>5
```

Explanation: This query starts from the vertex CS101 and repeatedly goes through all the vertices to which there is an ‘in’ edge with the label ‘prereq’ until it reaches one vertex which has 0 ‘in’ edges. There will be multiple paths in this case. Therefore we select the last path using the function tail() and output the length by counting number of vertices with count() function.

BONUS: Creating and initializing graph in two steps

```
gremlin> g = TinkerGraph.open().traversal()
==>graphtraversalsource[tinkergraph[vertices:0 edges:0], standard]
```

Explanation: We are opening a new graph and also defining a traversal source in one step.

```
g.addV().property(id, 1).as("CS101").addV().property(id, 2).as("CS201").addV().property(id,
3).as("CS220").addV().property(id, 4).as("CS420").
        addV().property(id, 5).as("CS334").addV().property(id, 6).as("CS681").addV().property(id,
7).as("CS400").addV().property(id, 8).as("CS526").
        addE("prereq").from("CS201").to("CS101").addE("prereq").from("CS220").to("CS201").addE("prereq").from("C
S334").to("CS201").
        addE("prereq").from("CS420").to("CS220").addE("coreq").from("CS420").to("CS220").addE("prereq").from("CS
681").to("CS334").
        addE("prereq").from("CS400").to("CS334").addE("prereq").from("CS526").to("CS400").addE("coreq").from("CS
526").to("CS400").iterate()
```

Explanation: Here we are adding all the vertices and edges at once. We can add vertices using addV() function and additional properties using property() method. Edges are added using addE() method which takes label as the parameter. The vertices between which the edge exists is defined by from() and to() functions.