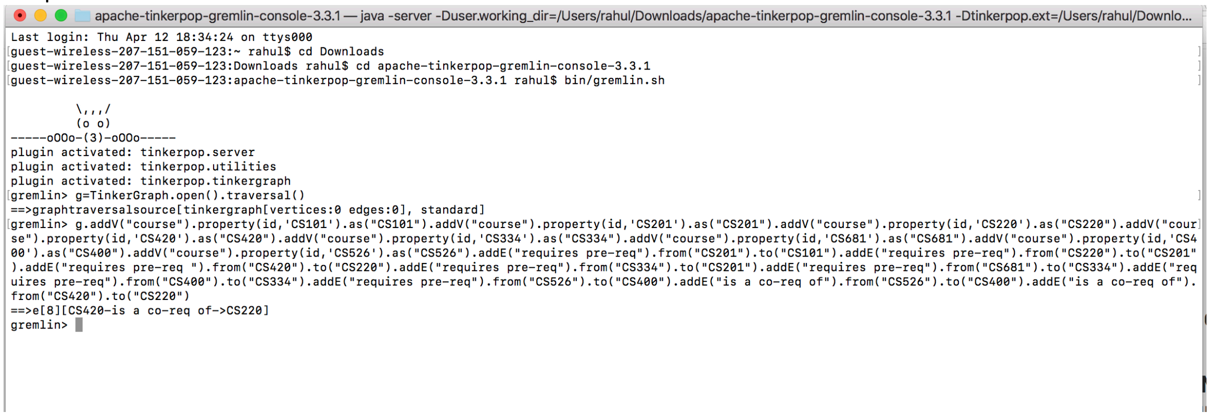
DATABASE ASSIGNMENT 4 – Bonus Queries

Name: Rahul Shreerang Lele **USC ID: 2483165273**

Query 1:

g=TinkerGraph.open().traversal()
g.addV("course").property(id,'CS101').as("CS101").addV("course").property(id,'CS201').as("CS201").addV("course").property(id,'CS220').as("CS220").addV("course").property(id,'CS420').as("CS420").addV("course").property(id,'CS334').as("CS334").addV("course").property(id,'CS681').as("CS681").addV("course").property(id,'CS400').as("CS400").addV("course").property(id,'CS526').as("CS526").addE("requires pre-req").from("CS201").to("CS101").addE("requires pre-req").from("CS220").to("CS201").addE("requires pre-req ").from("CS420").to("CS220").addE("requires pre-req").from("CS334").to("CS201").addE("requires pre-req").from("CS681").to("CS334").addE("requires pre-req").from("CS400").to("CS334").addE("requires pre-req").from("CS526").to("CS400").addE("is a co-req of").from("CS526").to("CS400").addE("is a co-req of").from("CS420").to("CS220")

Output:



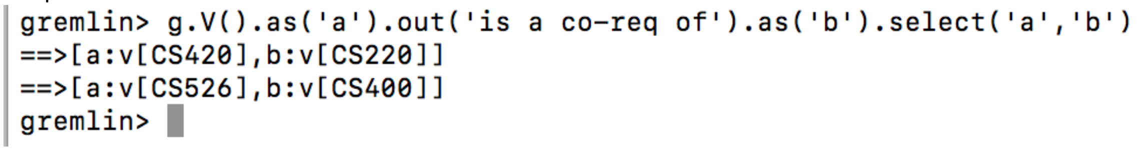
```
Last login: Thu Apr 12 18:34:24 on ttys000
guest-wireless-207-151-059-123:~ rahul$ cd Downloads
guest-wireless-207-151-059-123:Downloads rahul$ cd apache-tinkerpop-gremlin-console-3.3.1
guest-wireless-207-151-059-123:apache-tinkerpop-gremlin-console-3.3.1 rahul$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> g=TinkerGraph.open().traversal()
==>graphtraversalsource[tinkergraph[vertices:0 edges:0], standard]
gremlin> g.addV("course").property(id,'CS101').as("CS101").addV("course").property(id,'CS201').as("CS201").addV("course").property(id,'CS220').as("CS220").addV("cour
se").property(id,'CS420').as("CS420").addV("course").property(id,'CS334').as("CS334").addV("course").property(id,'CS681').as("CS681").addV("course").property(id,'CS4
00').as("CS400").addV("course").property(id,'CS526').as("CS526").addE("requires pre-req").from("CS201").to("CS101").addE("requires pre-req").from("CS220").to("CS201"
).addE("requires pre-req ").from("CS420").to("CS220").addE("requires pre-req").from("CS334").to("CS201").addE("requires pre-req").from("CS681").to("CS334").addE("req
uires pre-req").from("CS400").to("CS334").addE("requires pre-req").from("CS526").to("CS400").addE("is a co-req of").from("CS526").to("CS400").addE("is a co-req of").
from("CS420").to("CS220")
==>e[8][CS420-is a co-req of->CS220]
gremlin> 
```

Explanation:
The first statement creates an empty graph and sets a traversal for gremlin to traverse the graph. The second statement adds vertices and edges to the graph. The addV() statement adds vertices to the graph."course" is a label. The property() is used to set id for every vertex. The addE() statement adds edges to the graph. There are two types of edges, one have the label "requires pre-req" and other have the label "is a co-req of". from() is used to specify the source vertex and to() is used to specify the destination vertex.

Query 2:

g.V().as('a').out('is a co-req of').as('b').select('a','b')

Output:



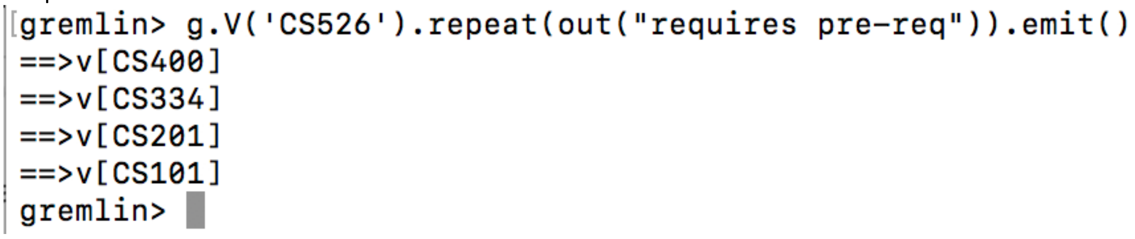
```
gremlin> g.V().as('a').out('is a co-req of').as('b').select('a','b')
==>[a:v[CS420],b:v[CS220]]
==>[a:v[CS526],b:v[CS400]]
gremlin> 
```

Explanation:
This query traverses the edges that have the label "is a co-req of" (doubly connected). The part of the query as('a').out('is a co-req of').as('b') selects vertices that have outgoing edges with the label "is a co-req of" and labels the source vertices as “a” and destination vertices as “b”. The select query outputs the vertices of the above considered edges.

Query 3:

g.V('CS526').repeat(out("requires pre-req")).emit()

Output:



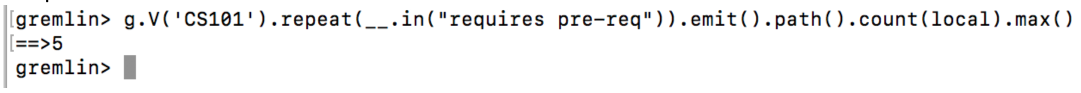
```
gremlin> g.V('CS526').repeat(out("requires pre-req")).emit()
==>v[CS400]
==>v[CS334]
==>v[CS201]
==>v[CS101]
gremlin>
```

Explanation:
The traversal begins at the vertex ‘CS526’. For Eg: The out("requires pre-req") traverses along the this outgoing labeled edge from CS526 to CS400. The repeat() is used to repeat this traversal for CS400 which gives next vertex CS334 and so on until there are no edges. The repeat() is like a loop to repeat the traversal for every vertex as it traverses. The emit() emits the vertices as it traverses.

Query 4:

g.V('CS101').repeat(__.in("requires pre-req")).emit().path().count(local).max()

Output:

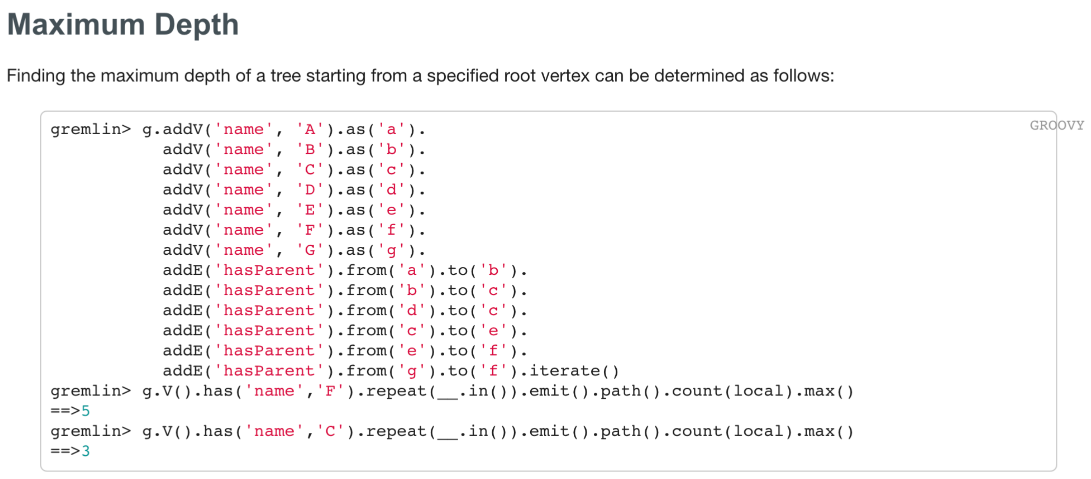


```
gremlin> g.V('CS101').repeat(__.in("requires pre-req")).emit().path().count(local).max()
==>5
gremlin>
```

Explanation:

The traversal beings at the vertex “CS101”.Repeat is used to get all the vertices having in going edges labelled ‘requires pre-req’. It uses path() to get all such paths. emit() gives out the paths. It uses count(local) to count the path length for each path and then used max() to select the longest path. For example, CS101 has 3 such paths. repeat(__.in(‘requires pre-req’)) repeats itself till the leaf is reached for each path. emit().path() emits each path and count(local) counts length each path and max() gives us path with maximum length.

Reference is taken from the following query from the Recipes tutorial:



Maximum Depth

Finding the maximum depth of a tree starting from a specified root vertex can be determined as follows:

```
gremlin> g.addV('name', 'A').as('a').
           addV('name', 'B').as('b').
           addV('name', 'C').as('c').
           addV('name', 'D').as('d').
           addV('name', 'E').as('e').
           addV('name', 'F').as('f').
           addV('name', 'G').as('g').
           addE('hasParent').from('a').to('b').
           addE('hasParent').from('b').to('c').
           addE('hasParent').from('d').to('c').
           addE('hasParent').from('c').to('e').
           addE('hasParent').from('e').to('f').
           addE('hasParent').from('g').to('f').iterate()
gremlin> g.V().has('name','F').repeat(__.in()).emit().path().count(local).max()
==>5
gremlin> g.V().has('name','C').repeat(__.in()).emit().path().count(local).max()
==>3
```

GROOVY