**Homework 4 CSCI -585**

1. **Gremlin command to create the graph**

graph=TinkerGraph.open()

g=graph.traversal()

vertex1= g.addV("CS101").property(id,'CS101').property("pre-req of","CS201”).next()

vertex2= g.addV("CS201").property(id,'CS201').property("pre-req of","CS220,CS334").next()

g.addE("requires pre-req").from(vertex2).to(vertex1).property(id,1)

vertex3= g.addV("CS220").property(id,'CS220').property("pre-req of","CS420").property("co-reqs","CS420"). next()

g.addE("requires pre-req").from(vertex3).to(vertex2).property(id,2)

vertex4=g.addV("CS420").property(id,'CS420').next()

g.addE("requires pre-req").from(vertex4).to(vertex3).property(id,3)

g.addE("is a co-req of").from(vertex4).to(vertex3).property(id,4)

vertex5=g.addV("CS334").property(id,'CS334').property("pre-req of","CS681,CS400").next()

g.addE("requires pre-req").from(vertex5).to(vertex2).property(id,5)

vertex6=g.addV("CS681").property(id,'CS681').next()

g.addE("requires pre-req").from(vertex6).to(vertex5).property(id,6)

vertex7=g.addV("CS400").property(id,'CS400').property("pre-req of","CS526").property("co-reqs of", "CS526").next()

g.addE("requires pre-req").from(vertex7).to(vertex5).property(id,7)

vertex8=g.addV("CS526").property(id,'CS526').next()

g.addE("requires pre-req").from(vertex8).to(vertex7).property(id,8)

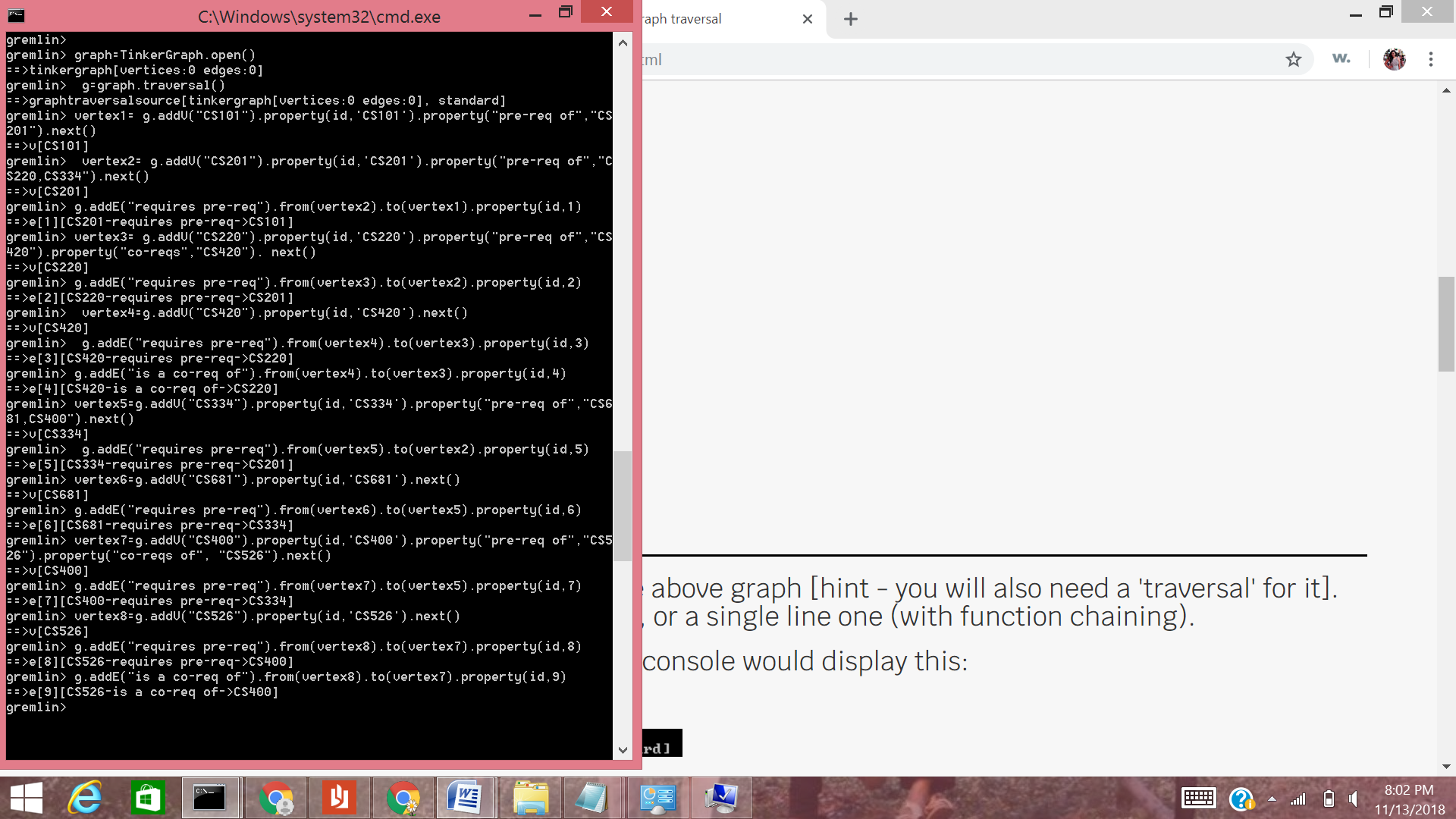
g.addE("is a co-req of").from(vertex8).to(vertex7).property(id,9)

**Explanation**: TinkerGraph.open() is used to create a graph with no edge or vertice currently. Then we will use g as a traversal. I have created 8 vertex names from vertex1 to vertext8. Each is created using .addV()

.property() is used to assign properties . For each vertex, I have added id as property and also ‘requires pre-req’ or ‘pre-req of’, depending on the relationship of the vertex with other vertices.

I have used .addE(name).from(‘vertexname’).to(‘vertexsecondname’).property() to created edges using .addE() with the name specified and this edge will be from ‘vertexname’ to ‘vertexsecondname’ and will have the property mentioned (requires pre-req or is a co-req of)

I have created edges with names ‘requires pre-req’ for the vertex having pre requisites and ‘is a co-req of’ for the double connected vertex



1. **Write a query to output the double connected nodes**

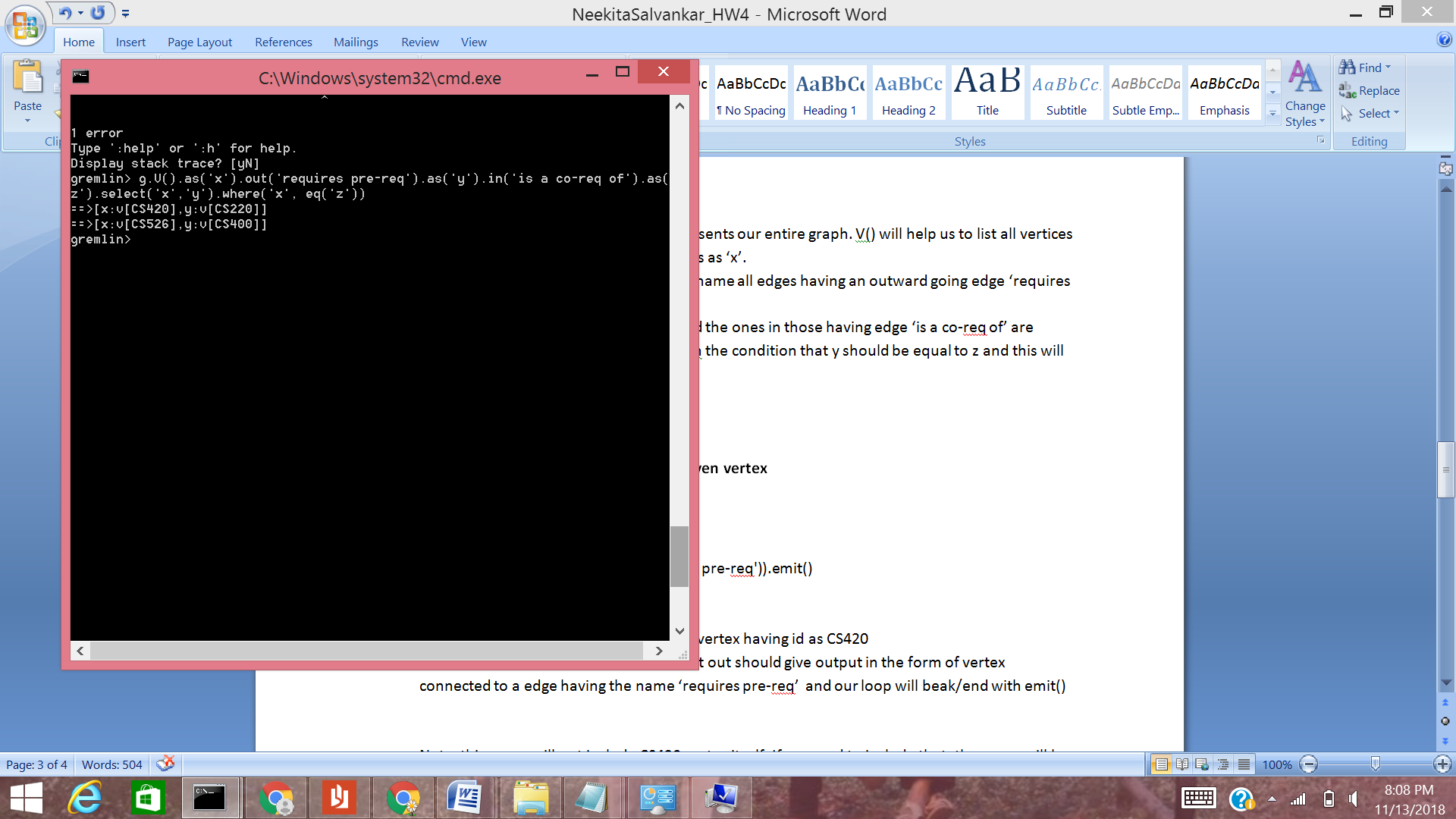
g.V().as('x').out('requires pre-req').as('y').in('is a co-req of’).as(‘z’).select(‘x’,’y’).where( ‘x’,eq(‘z’))

Explanation:

We are using g as a traverser and it represents our entire graph. V() will help us to list all vertices in the graph and all of them will be names as ‘x’.

We use .out('requires pre-req').as(‘y’) to name all edges having an outward going edge ‘requires pre-req’ as ‘y’.

Now these ‘y’ vertices are considered and the ones in those having edge ‘is a co-req of’ are named as ‘z’. We then select x and y with the condition that y should be equal to z and this will help us get the double connected nodes.



1. **Query to output all the ancestors of a given vertex**

ssI have taken CS420 as an example

g.V().has(id,'CS420').repeat(out('requires pre-req')).emit()

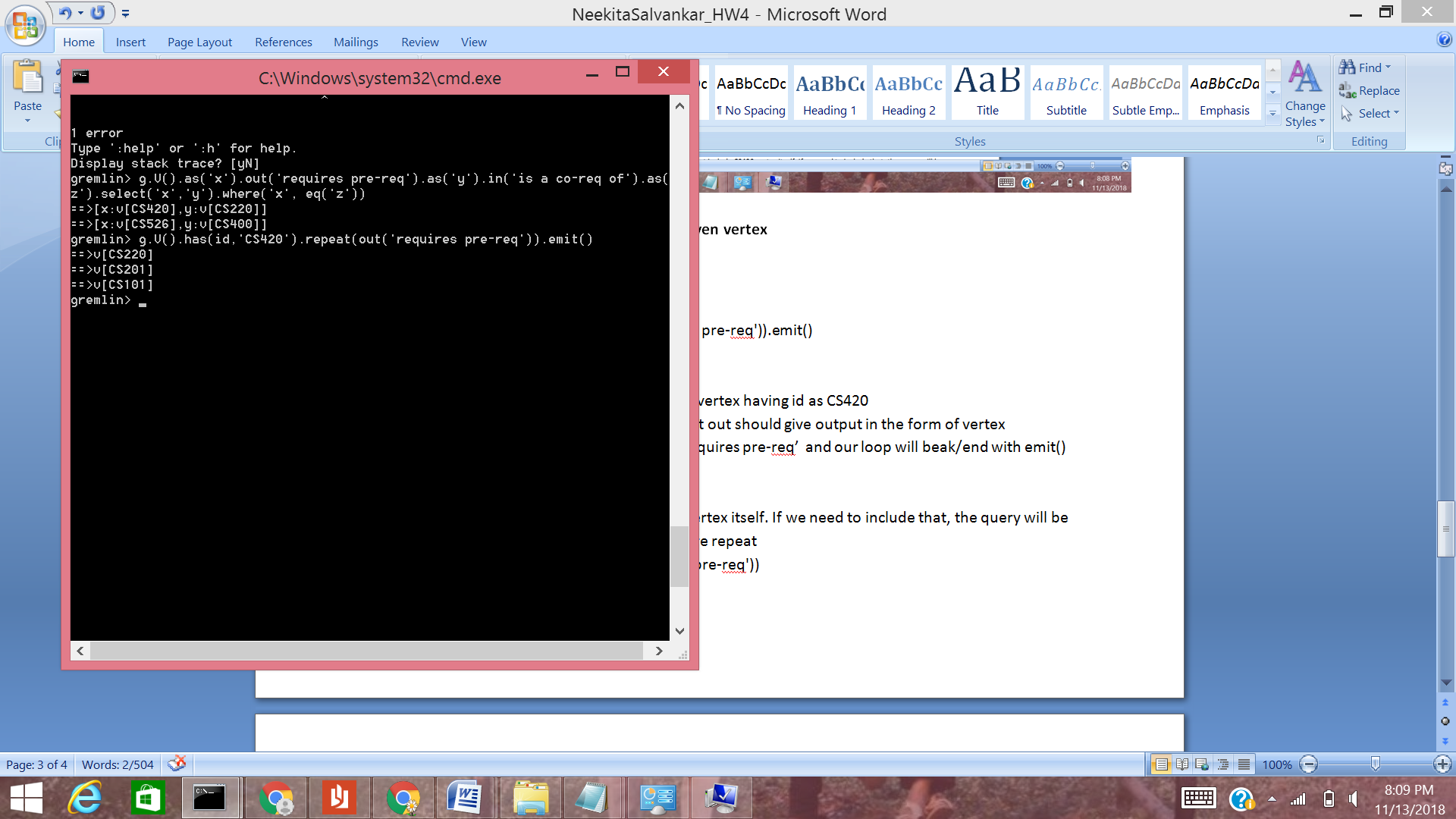
**Explanation**: g is used to traverse

.V().has(id, ‘CS420’) is used to access the vertex having id as CS420

I have used repeat with the condition that out should give output in the form of vertex connected to a edge having the name ‘requires pre-req’ and our loop will break/end with emit()

Note: this query will not include CS420 vertex itself. If we need to include that, the query will be modified a little, that is, shift emit() before repeat

g.V(‘CS420’).emit().repeat(out('requires pre-req'))



1. **Query to output max depth from a given node**

Taking the start node as CS201

g.V().has(id,'CS201').emit().repeat(\_\_.in('requires pre-req')).tail(1).path().unfold().count()

**Explanation:**

g is the traverser representing our graph and g.V() will list all the vertices.

emit() is used immediately to emit and use CS201 immediately as we require it to calculate the overall depth. After emitting CS201, it will enter the loop of repeat. The repeat condition requires verticies having incoming edges with label ‘requires pre-req’

I have used \_\_in() for getting a top to bottom path and path() helps in getting the same(first vertex to last vertex)

Count() will count the number of lines, that is, the depth and unfold will help to display vertices line by line

