Q1. For making a graph we should use built-in function like addvertex and addedge. We can do everything we want with these functions as they can get different kind of parameters but for this homework we just need to add a label which is name to every edge and also to every vertex. For example “cs526 = graph.addVertex('name','cs526')” this query will make a vertex in our graph with label of ‘cs526’ and we can call this node just by refereeing its name.

=====================================================================================

Q2. For second query we should define extra edges to our graph, somehow makes it directed! But with different label for backward edges(just for two pairs of course not all of them).

“g.V().match( \_\_.as('a').out('pq').as('b'), \_\_.as('b').out('aa').as('a')).select('a','b').by('name')”

This query takes every edge that has outgoing edge with the name of ‘pq’ and also edges with name of ‘aa’(out() will Get the out adjacent vertices to the vertex.) As we have just these edges for two pair of our courses it will give us those courses. It assume names ‘a’ and ‘b’ as an variables for getting those edges and returning them. Finally ‘select’ statement would give us those results(name of them actually)

=====================================================================================

Q3. For the third part the following query would give us ancestors of the given node until the root. This query repeat for every node that has ‘pq’ edge until it has not any outgoing edge and for every loop it prints out its result. Why it does not show us the first course? Because If until() comes after repeat() function it is like a do/while looping.

“g.V(cs526).repeat(out('pq')).until(outE().count().is(0)).emit().values()”

=====================================================================================

Q4. The last one, will compute the longest possible depth in out tree. Our traversal begins at a particular node and comes down for every vertex that has child with the incoming edge of ‘pq’. Then find the maximum between these path that were ended to the leafs.

“g.V().has('name','cs101').repeat(\_\_.in('pq')).emit().path().count(local).max()”