Yong Zhou
Hw_3

Question 1:

- Description: As a graph of this situation, Dirk to be the first node of the chain, after Dirk passes out the message, at most saw by 7 friends, will pass back to Dirk. Otherwise, this chain of at most 7 friends will not exists. with n nodes(person)
    - M from Dirk to first layer, (from Dirk to all possible directions, adding friends one "layer")

- Algorithm:

```
Initialization:
        message M;
        Dirk <- first node
        A <- array list
        A[0] = Dirk
        Layer <- L
Run BFS (from the textbook):
    If   first layer L1 to layer L4 is not empty:
        For i in L1 to L4:
                For person in i:
                        if i = L4 and any one person has at least two friends in L3.
                                Return chain7exist

                        Else i <= L3  and any one person has at least one friends in same
    layer or at least two friends in the previous layer
                                Return chain7exist
                        Endif
         endfor
    Else:
        Return Not exist
    endif
```

- Running time O(m+n):
    - If there are n persons, and m layers, previous one person connect to next layer pass message take time m, went throuht n persons. Run BFS takes O(n+m). When two persons in array A, and pass the message back which require another direction of any two nodes. Which is 2* O(m+n). Thus, total time is O(m+n).

- Correctness:
    - Run BFS, to satisfy a chain at most 7 friends that is Dirk get back the message in 8 person or shorter than 8 persons. By BFS definition, friends i and j will at same layer or next layer layers or previous layer. The maximal layers in this algorithm is 4 layers, that is friends with at least two friends in previous layer and pass M to

backward to Dirk. If two persons are friends in 4th layers, pass back M to Dirk use the shortest path will be 4 + 4 = 8 by BFS.

- Prove:
  - By contradiction:
    - Let us say there is a chain at most 7 friends in the layer greater than 4 or two persons are friends in the same layers of 4. From textbook p80, BFS is computing shortest paths to the nodes that s(starting node) can reach. If M from Dirk to a person who has friend at layer 5th or more, it does not possible to reach back to Dirk in 3 edges. Also, if two persons in the 4th layers are friends. when M is baking to Dirk that at least takes 4 steps, which the chain will greater than 7 friends.

Question 2:

Algorithm:

- Initialize:
  - S <- empty stack
  - u<- starting vertex
  - v<- out-neighbor of visiting v

For each vertex u of the graph, mark u as unvisited.
For each vertex u of the graph do Visit(u) recursively:
    If u unvisited
        Mark u as visited
        For each out-neighbor v of u, do Visit(v)
        Prepend u to S
For each element u of S in order, do assign (u, u) where assign (u, root) is the recursive subroutine.
    If u has not assigned to a component:
        Assign u as belonging to the component whose root is root
        For each in-neighbor v of u, do assign (v, root)

- complexity:
  - O (v + e) due to it using adjacency list. Reversing a graph is O (v +e). the total running time is O (v + e), v is the total vertex and e is the total edges.
- Correctness
  - For every iteration of visited(u), that is use DFS, until to get no out-neighbor and put in the stack of (u), when transposing reverses the direction of all edges, node G^t has the same strong component visited by the DFS.
  - Since G and G^t have the same strong components with the same stack orders of all nodes,
- Prove:
  - In contradiction. Assume the algorithm could not find the strong connection component. Either the stack orders of all nodes will go wrong or the DFS process is not fit on graph G.
  - First of all, the stack is filled by any node in G, and followed by its out-neighbor, by DFS from textbook(p.) until all vertices are reachable from the DFS starting

point, and put in stack, when stack pop out the node and run DFS by reversed graph, the edges that connect two components are revered. Therefore, the order from the stack must be the same. Therefore, the assume is not success.