Assignment 3.

Name: Guangfeng Lin
Student #: 301312131

Question 1

pseudocode of algorithm for transpose an adjacency list.

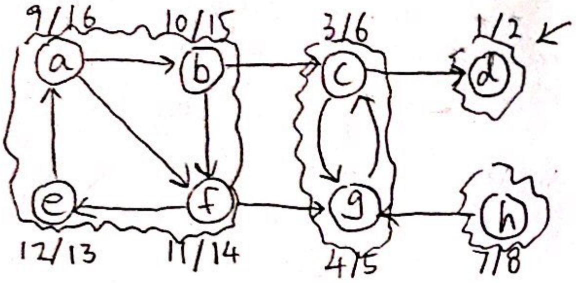
```
res(G)
    Initializing Graph G_transpose
    for i=0 to size V[G]
        for j=0 to size of G[i]
            add(G_transpose, j)          O(V+E).
    Return G_transpose
```

Explanation: First, initializing a empty graph call G_transpose, then using a nest for loop to add the transpose vertices to the new G_transpose graph. The complexity of this algorithm is $O(V+E)$.

Question 2.

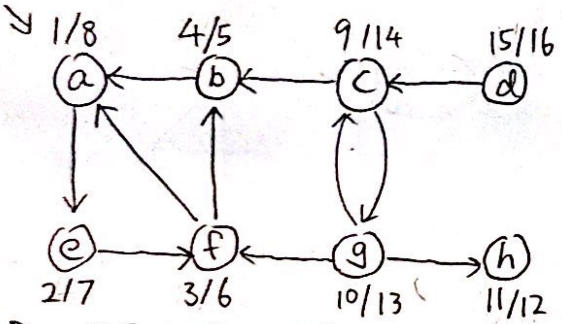
By the definition of strongly connected component of a directed graph G, that u,v∈G is strongly connected if every two vertices are reachable from each other. Thus, for every vertices u, v ∈ G, there is a path from u to v and v to u.

Example:



SCC = {a, b, e, f}
{c, g}
{d}
{h}

DFS for G with this order (d, c, g, h, a, e, f, b)



Do DFS for G^T first

Let S(v) = {c, g}, S(u) = {a, b, e, f}, we can see vertex C is reachable from vertex e by the path e, a, b, C. Therefore, u can reach v or v is reachable from u if and only if this two strongly connect component is connecting to each other. //

Question 3.

Pseudocode of algorithm

case 1: $e \in T$

If a edge's weight increase, We can remove that edge and separate the mst into two subtree as tree1 and tree2.

```
New_MST (G, e, w){
    tree1_vertex = BST (G, u); // u is the endpoint of subtree tree1.
    tree2_vertex = BST (G, V); // V is the end point of subtree tree2.
    for each edge in G {
        min = w;
        if (starting_edge == tree1_vertex && ending_edge == tree2_vertex){
            if (edge_weight < min) {
                min = edge_weight; // find out the new minimum edge between
                                   // subtree tree1 & tree2.
            }
        }
    }
}
```

Case 2: $e \notin T$

If the weighe increased edge is not in the tree then the tree is stay unchanged.

For case 1, we separate the tree into two subtrees, then we search the new minimum weighted edge to connect these two subtrees.

Question 4.

Remove the negative edge from G and state the new graph as G'.
If the result Dijkstra's (G', s, t) + w(u,v) (w(u,v) is the weight of the missing edge) is negative then the graph contains a negative cycle, thus shortest path cannot be calculated. Otherwise, we can just use the modified Dijkstra's algorithm below.

modified Dijkstra's algorithm:

```
M_Dijkstra(G', s, t)
    return min (Dijkstra (G', s, t), (Dijkstra (G', s, u) + w(u, v) + Dijkstra(G', v, t)))
    // This will output a shortest path for G'
```

Explanation: If the shortest path does not contain the negative edge w(u,v) then we can just use Dijkstra (G', s, t) to calculate the shortest path from s to t.

If the shortest path contains the negative edge w(u,v) then we can calculate the shortest path by doing M_Dijkstra (G', s, u) to calculate the shortest path from vertex s to u first, and then doing M_Dijkstra (G', v, t) to calculate the shortest path from vertex v to t and we add w(u,v) to the result.