CSE 2321 Homework 9 Template

Problem 1

```
DFSModified(G)
    boolean hamiltonian = false
    node s = u # s is the start node
    for each u in V
        u.color = white
        u.p = nil    # p is for parent
    for each u in V
        if u.color == white
            hamiltonian = Visit(G,u, s)
        if u.color == black
            u.color == white # revisit past nodes
    return hamiltonian

Visit(G, u, s)
    boolean hamiltonian = false
    for each v in G.Adj[u]
        u.color = gray
        if v.color == white
        v.p = u
        Visit(G, v)
        u.color = black
        if v == s # If the next node is the starting node
            hamiltonian = true
```

Problem 2

```
FindSink(G)
    int i, j = 0
    boolean hasSink = false
    for i to |V| && !hasSink
        if !(i == j) && G[i][j] == 0
```

```
        # j is not a sink, next column
            j++;
            hasSink = isSink(i)
        else if !(i == j) && G[i][j] == 1
         # i is not a sink, next row and column
            i++
            j++
            hasSink = isSink(j)
        else if i > |V| || j > |V|
            hasSink = false
    return hasSink

isSink(G, x)
    boolean isSink = true

    for int i = 0 to |V|
        # The xth row should be all 0
        if G[k][i] != 0
            isSink = false
        i++
    for int i = 0 to |V|
        # The xth column should be all 1s
        if i != k && G[i][k] != 1
            isSink = false
        else if !(i == j) && G[i][j] == 1
            isSink == false
        i++
    return isSink
```

Problem 3

```
Diameter(G)
    int max = 0 # the diameter
    for int i = 0 to |V| # test all starting points
        # check all distances to other vertices
        for int j = 0 to |V|
```

```
            int length = BFSModified(G, i)
            # Note: if two nodes are not connected, infinity
            if length != infinity && length > max
                max = length
            j++
        i++

    return max

BFSModified(G, s, e)
    for each v in V
        v.dist = infinity
        s.dist = 0
        q.enqueue(s)
        while q != 0
            v.dequeue
                for each w s.t. (v,w) in E
                    if w.dist = infinity
                        q.enqueue(w)
    return w.dist # Modified to return length of path
```

The running time of this algorithm is $O(|V|^2 + |V||E|)$. BFSModified has the same running time as BFS, which is $O(|V| + |E|)$. The for loops have the running time of $O(|V|^2)$.