CSCD320 Homework 7, Winter 2012, Eastern Washington University. Cheney, Washington.

Name: Eric Fode **EWU ID:00530214**

Solution for Problem 1

1. BFS Output: D,B,I,N,M
2. DFS Output: D,B,N,M,I

Solution for Problem 2

1. Pseudocode:

getNextPerm($graphList$)

Input: A graph represented as an adjacency list
Result: A graph represented as a matrix

```
/* create a matrix of the proper size with all 0 values          */
matrix = int[graphList.V.size][graphList.V.size];
for i ← 0 to graphList.V.size do
    edgeNodecur = graphList.V[i];
    while graphList.V[i]! = null do
        matrix[i][edgeNode.outherNode] = 1;
        edgeNode = edgeNode.next;
return matrix;
```

2. Analysis of running time:$O(E)$

Solution to problem 3

1. Algorithm:

   1. Set the starting point of the search and push it onto a stack s
   2. Pop item i off s add to list l of visited verts
   3. Push the adjacent verts of l.last using adjacency matrix $O(V)$
   4. Goto 2 if s is not empty

2. Problem: When using an adjacency matrix you have to iterate through every vertex to find the adjacent vertices. This is orders more expensive then just having the information you are looking for listed (like in the adjacency list). This is because matrix involves for every node a search and the list does not.
3. Time complexity: $O(V^2)$ as compared to $O(|V| + |E|)$