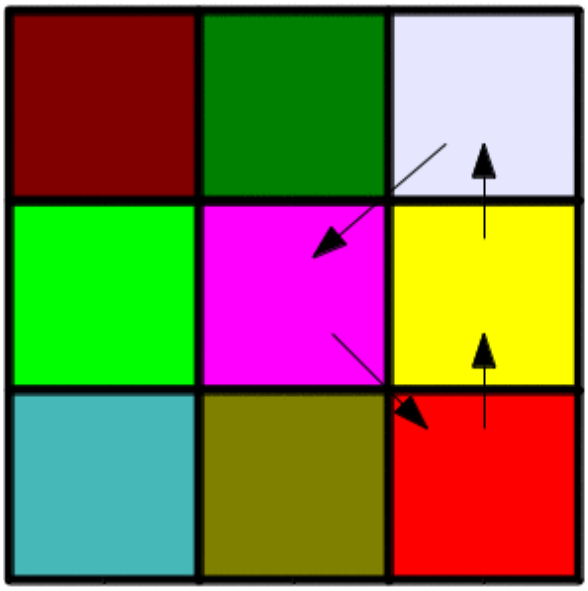
CS5800 Algorithms Spring 2014 Assignment #2

1. (25 points) Widdernish is a game played with a 3x3 grid of colored squares like this:

Each move of the game consists of shuffling the squares counter-clockwise on one side of the grid together with the center square like this:



A

B

where the right side of the square of the grid was shuffled in grid A above to produce grid B. The object of Widdernish is to rearrange the squares so that they are in a target arrangement. Here is an example of such a goal:

Using pseudo-code, develop an algorithm that determines how to rearrange the the squares using valid moves so as to accomplish the goal, if the goal is possible. Your algorithm must work for an arbitrary initial arrangement of the squares and arbitrary goal arrangement.

2. (15 points) Run the DFS algorithm on the following undirected graph, showing each step. When there is a choice for the next vertex, use the one that is smaller numerically.

Edge	Edge	Edge	Edge
{1,5}	{1,6}	{1,8}	{2,3}
{2,5}	{2,7}	{3,4}	{3,7}
{3,9}	{4,5}	{4,6}	{4,7}
{5,8}	{5,9}	{5,10}	{6,8}
{7,9}	{8,10}	{9,10}	

3. (20 points) Perform depth-first search on each of the following graph; whenever there is a choice of vertexes, pick the one that this numerically smaller. Classify each edge as a tree edge, forward edge, back edge, or cross edge, and give the pre and post number of each vertex. Determine whether the graph is acyclic. If the graph is acyclic, topologically sort it.

Edge	Edge	Edge	Edge
(1,6)	(1,8)	(1,9)	(2,3)
(2,5)	(2,7)	(3,7)	(3,9)
(3,10)	(4,5)	(5,10)	(6,2)
(6,4)	(7,5)	(7,6)	(7,9)
(8,6)	(8,7)	(8,10)	

4. (20 points) In a social networking site one has (directed) friendship links. They are represented

using adjacency lists. Define an acquaintance as a friend of a friend. Someone can be an acquaintance or yours more than once, and you can be an acquaintance of yourself (also more than once). Develop an algorithm for computing the number of acquaintances of each person (including multiplicities). Your algorithm must run in linear time, and you must prove that it runs in linear time.

5. (20 points) Develop an algorithm for finding cycles. This algorithm should have two parameters: a directed graph and a node. It must find a cycle containing the specified node or determine that there is no cycle containing the node. Your algorithm must run in linear time, and you must prove that it runs in linear time. Perform your algorithm on the following graph and the node labeled "4":

Edge	**Edge**	**Edge**	**Edge**
(1,6)	(1,8)	(2,3)	(2,5)
(2,7)	(3,7)	(4,5)	(5,2)
(5,10)	(6,2)	(6,4)	(7,5)
(7,6)	(8,6)	(8,10)	(9,2)
(9,3)	(9,8)	(10,3)	(10,8)