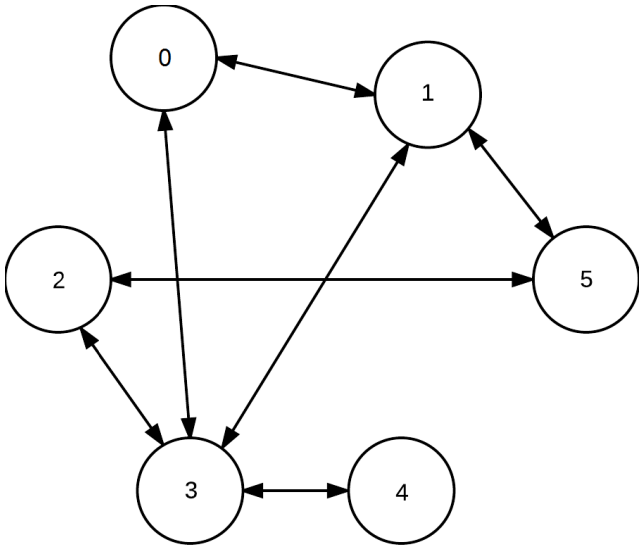
Labwork 6

Your work will be evaluated using Visual Studio 2015. You are not allowed to make any changes to the file "graph.h".

Q1: You are given a missing implementation of a Graph class. Implement the following functions in order to complete the class. Examples are provided in Q2.

- **int indegree(int n):** Returns the in-degree of node *n*. The in-degree of a node is equal to the incoming connections to that node.
- **int outdegree(int n):** Returns the out-degree of node *n*. The out-degree of a node is equal to the outgoing connections from that node.
- **int num_edges():** Returns the total number of edges in the graph. Note that the graph class you are given represents directed graphs; therefore you need to count bidirectional edges as 2.

Q2: Write a main function which creates a graph object which represents the graph below:



In order to create the graph object; you need to declare a two-dimensional array which represents the adjacency matrix. In this matrix, element (i, j) should be 1 if there is a connection from *i* to *j*; and 0 if not.

For example, for the graph above, adj[0][3] should store 1, since there is a connection from node 0 to node 3.

Similarly, adj[1][1] should store 0, since there is no connection from node 1 to node 1.

After creating the graph object, write necessary code to:

- Print in-degrees of all nodes in the graph
- Print out-degrees of all nodes in the graph
- Print total number of edges
- Print the path of nodes visited with DFS when starting from node 2.
- Print the path of nodes visited with BFS when starting from node 5.
- Print the shortest path from node 5 to node 4.

Remember that you already implemented the necessary functions for the first three items; and functions for the last three items are already given to you.

Here are some example outputs to help you evaluate your work:

```
In-degree of node 4:        1
Out-degree of node 3:       4
Number of edges:            14
DFS from node 4:            4 3 2 5 1 0
BFS from node 5:            5 1 2 0 3 4
Shortest path from 5 to 4:  5 1 3 4
```