```
CS 6301.002.  Implementation of advanced data structures and algorithms
Spring 2016;  Wed, Apr 6.
Long Project 5: Matching

Ver 1.0: Initial description (Wed, Apr 6, 11:00 AM).
Max excellence credits: 1.2.
Due: 11:59 PM, Sun, May 1 (1st deadline), Sun, May 8 (2nd deadline).
```

Code base

Java library, java examples discussed in class by instructor. Do not use code from outside sources.
Starter code: Graph/Vertex/Edge classes, LP5Driver.java.

Project Description

In this project, you will implement 3 algorithms for matching. Level 1 is required, and levels 2 and 3 are optional.

Level 1 (EC: 0.2)

Implement the algorithm discussed in class to find a maximum cardinality matching in a given bipartite graph G=(V,E). You can make modifications to make it more efficient, but do not implement some other algorithm (like maximum flow) and use that to find a matching. Use BFS to test if the input graph is bipartite: partition the nodes into V = X U Y (outer and inner nodes), by placing nodes in even layers in X, and nodes in odd layers in Y. All edges must connect a node in X to a node in Y. If any edge of the graph connects 2 nodes of X, or 2 nodes of Y, then the graph is not bipartite. If the graph is not bipartite, then output the message "G is not bipartite" and exit. Otherwise, your program should run the maximum matching algorithm discussed in class and output a maximum matching. Excellence credit will be awarded based on quality of the code, and improved running time obtained by identifying multiple augmenting paths each time the alternating forest is built. Submit a driver program **LP5Lev1.java**.

Level 2 (Optional; EC: 0.5)

Implement the algorithm discussed in class to find a maximum cardinality matching in an arbitrary graph G=(V,E). Your program outputs the cardinality of a maximum matching in the graph, and if VERBOSE is true, output the edges of the matching. Submit a driver program **LP5Lev2.java**.

Level 3 (Optional; EC: 0.5)

Implement the algorithm discussed in class to find maximum-weight matchings in bipartite graphs. Submit a driver program **LP5Lev3.java**.

Input specification

An undirected graph is given as input, in the format expected by readGraph method. If there is a command line argument, input is read from that file. Otherwise, read input from the console. Do not process lines in the input after the input graph has been read. A second command line argument, if present, sets the boolean VERBOSE to true.

Output specification

For each problem, print the size of the matching (cardinality of maximum matching for levels 1 and 2, and the weight of the matching for level 3). If the boolean VERBOSE is true (by passing a second command line argument when the program is run), then output the edges of the matching, one per line.

Sample inputs and outputs

```
Sample input:
5 4
1 2 2
3 2 6
3 5 2
4 2 2

Output for levels 1 and 2 (if run as "java LP5Lev1" or "java LP5Lev2", with input pasted on console):
2
```

```
Verbose Output for levels 1 and 2 (if run as "java LP5Lev1 in1.txt true"; input is in file in1.txt):
2
1 2 2
3 5 2

Output for level 3:
6

Verbose output for level 3:
6
3 2 6
```