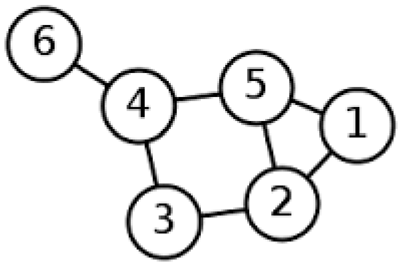
CPS112: Computational Thinking with Data Structures & Algorithms
Lab 12: Priority Queue and Graph Basics

1. Minimum Priority Queue basics
   Assume you are starting with an empty priority queue. Show the tree **AND** array representation of the priority queue after each step given below:
   (a) enqueue 12
   (b) enqueue 23
   (c) enqueue 9
   (d) enqueue 8
   (e) enqueue 10
   (f) enqueue 15
   (g) dequeue
   (h) dequeue

2. Provide the both the adjacency list and adjacency matrix representations for the graph below.



– MIDWAY CHECK –

3. In class we discussed and developed code for the Minimum Priority Queue `enqueue` operation. For this problem you will write the code to `dequeue` the item with the minimum priority. When the item is removed, the minimum heap properties must be maintained. This is accomplished by taking the last item in the list, make it the root, and percolate it down into the proper position. You should review your notes and the example you worked through above before writing the code. You will need to fill in the following methods:
   (a) `dequeue`: Gets the item at the root, sets the last item to be the root and percolate down. Be sure to consider what happens if dequeing would leave you with an empty tree.
   (b) `percolateDown`: swaps node down in the tree with minimum priority child until node is in correct position
   (c) `getMinChild`: Helper function to determine which (if any) of at most two children nodes are the minimum. Returns the index position of the minimum priority child or null if there is no child.

4. In your *BinaryTree.java* from last week, add a new method call `countLeaves`. It should take no parameters and return the number of leaves in the tree. Modify the main method to call `t8.countLeaves()` to test your implementation.

– FINAL CHECK –