Assignment 4: Breadth First Search Using a Circular Linked List Implementation of a Queue
Name: Jimi Andro-Vasko
Date: 3/25/14

Program Input/Output:

- An integer is read first which will be the number of nodes in the graph
- Each subsequent row is read and each row has 2 integers which represent an edge from the first vertex to the other
- The output will be the out-neighbor list for all the vertices
- The program will also output all the vertices in breadth first search order
- Only integers will be read and the first integer which indicates the number of nodes will not be greater than 100

High Level View:

- The number of nodes where read and an out-neighbor list was created
- Then starting from vertex 0, the program marked all the out-neighbors and enqueue them
- When a value was dequeued, all of its non marked out-neighbors are then marked and enqueued
- The program terminates once the queue is empty

Implementations/Specifications:

- A circular linked list class was implemented to represent a circular queue where each node was a dynamically allocated struct
- The out-neighbor list was implemented using an array of structs in which there was a pointer field that pointed to the next vertex in the out-neighbor list
- There was a constructor and two methods that were used to manipulate the queue, they were enqueue and dequeue and there was a print method which outputted the out-neighbor list
- The enqueue and dequeue methods maintained a dummy node for the circular queue implementation to avoid having to deal with an extra tail pointer
- The control of how values where enqueued and dequeued and the overall implementation of breadth first search were all done in int main

Compiling/MakeFile/Execution:

- No make file was used in this assignment
- The GNU Linux Compiler was used to compile the program using g++ bfs.cpp
- The output file generated by default was a.out and ./a.out < input.txt where input.txt is the input file which was read by using linux redirection